

(51) Int.Cl. ⁷	識別記号	F I	テラコート* (参考)
G 0 6 F 17/30	1 8 0	G 0 6 F 17/30	1 8 0 E
12/00	5 4 7	12/00	5 4 7 A

審査請求 未請求 予備審査請求 有 (全532頁)

(21) 出願番号 特願平8-516356
 (86) (22) 出願日 平成7年11月13日(1995.11.13)
 (85) 翻訳文提出日 平成9年5月12日(1997.5.12)
 (86) 国際出願番号 P C T / U S 9 5 / 1 5 0 2 8
 (87) 国際公開番号 W O 9 6 / 1 5 5 0 1
 (87) 国際公開日 平成8年5月23日(1996.5.23)
 (31) 優先権主張番号 0 8 / 3 3 9 , 4 8 1
 (32) 優先日 平成6年11月10日(1994.11.10)
 (33) 優先権主張国 米国 (U S)
 (31) 優先権主張番号 0 8 / 5 2 7 , 1 6 1
 (32) 優先日 平成7年9月12日(1995.9.12)
 (33) 優先権主張国 米国 (U S)

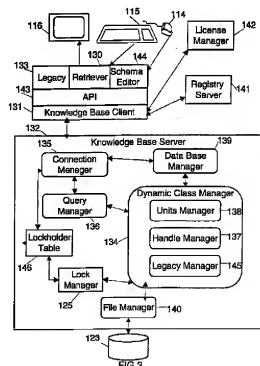
(71) 出願人 ケイティス、インク
 アメリカ合衆国カララドウ州80302、ボウルダ、トウェンティシックス・ストリート 1909番
 (72) 発明者 カヴァナー、タマス、エス
 アメリカ合衆国カララドウ州80302、ボウルダ、ベルヴー・ドライブ 65番
 (72) 発明者 ビオール、クリスタファ、ダブルユー
 アメリカ合衆国カララドウ州80302、ボウルダ、ケーガ・ドライブ 879番
 (74) 代理人 弁理士 真田 雄造 (外2名)

最終頁に続く

(54) 【発明の名称】 オブジェクト指向データベース管理システム

(57) 【要約】

本発明は、オブジェクト指向データベース管理システムのための方法及び装置を提供する。本発明は、知識ベースクライアント及び知識ベースサーバー(132)から成るクライアント/サーバーアーキテクチャーにおいて有利に使用することができる。複数のユーザーが同時にこのシステムにアクセスすることができる。好適具体例において、知識ベースサーバー(132)は、ダイナミッククラスマネージャ(134)、接続マネージャ(135)、質問マネージャ(136)、ハンドルマネージャ(137)、ユニットマネージャ(138)、データベースマネージャ(139)、及びファイルマネージャ(140)を備えることができる。オブジェクト指向データベースシステムは、ヒエラルキーである。知識ベースにおける各インスタンスは、クラスのメンバーであり、かつクラスは、親クラスのサブクラスであり、そして、以下同様にすることができる。



【特許請求の範囲】

1. ペアレント・チャイルド／クラス・サブクラス構造で組織化されたイグジスティング・インスタンスについての特徴を記述している情報のオブジェクト指向表現であって、インスタンスの本質的表現は、ペアレント・クラスからアトリビュートを受け継いでいることに加え、そのインスタンスが属するクラスから局部的に得られる情報に依存しているものと、

誘導され相互に影響を及ぼす方法で上記オブジェクト指向表現をただす手段と、

サーチ結果を表示する手段と、
表示手段によって表示される特徴と関連した特定の情報を選択する手段とを備えたデータベース・マネージメント・システム。

2. 上記情報のオブジェクト指向表現における1つのクラスは、少なくとも、ハンドルを持ったクラス・オブジェクトとして表現され、このクラス・オブジェクトはそのペアレント・クラスであるとするペアレント・ハンドルを持つと共に、サブクラス・リストを持っており、このサブクラス・リストは上記クラス・オブジェクトのサブクラスであるとする一連のクラス・ハンドルを備え、またこのクラス・オブジェクトはそのアトリビュートであるとするために用いられるハンドルのリストを備えたアトリビュート・リストを含むと共に、サブトリー・インスタンス・カウントを含んでおり、このサブトリー・インスタンス・カウントは上記クラス・オブジェクトに属しかつ上記クラス・オブジェクトの全ディセンダントの中にあるインスタンスの総数の示度数でなる請求項1記載のデータベース・マネージメント・システム。

3. 上記情報のオブジェクト指向表現における1つのクラスは、少なくとも、ハンドルを持った次位のクラス・オブジェクトとして表現され、この次位のクラス・オブジェクトはそのペアレント・クラスであるとするペアレント・ハンドルを持つと共に、ローカル・リストを持ち、このローカル・リストは上記次位のク

ラス・オブジェクトに属するインスタンスであると確認するために用いられ得る一連のハンドルを備え、またこの次位のオブジェクトはそのアトリビュートであ

るとするために用いられ得るハンドルのリストを備えたアトリビュート・リストを含むと共に、サブトリー・インスタンス・カウントを含んでおり、このサブトリー・インスタンス・カウントは上記次位のクラス・オブジェクトに属しかつ上記クラス・オブジェクトの全ディセendantの中にあるインスタンスの総数の示度数でなる請求項2記載のデータベース・マネージメント・システム。

4. 上記サーチ結果を表示する手段は、ユーザがサーチや上記クラスについて総てのディセendantで動かししているところのクラス中に存在するインスタンスの総数の示度数を与えるサブトリー・インスタンス・カウントに応じて、数値を表示する手段を備えてなる請求項3記載のデータベース・マネージメント・システム。

5. 上記インスタンスは、不確定な特徴用として割り当てられた記憶装置を追加することなしで、情報の所属するクラスとリストとして表現されてなる請求項1記載のデータベース・マネージメント・システム。

6. イグジスティング・データを、ペアレント・チャイルド/クラス-サブクラス構造で組織化されたイグジスティング・インスタンスについての特徴を記述している情報のオブジェクト指向表現を有する階層制オブジェクト指向シェーマへ、容易に組織化する承継物手段を更に備えてなる請求項3記載のデータベース・マネージメント・システム。

7. 上記承継物手段は、内容、形式において、インスタンスの型の機能としてインスタンスの記述を標準化するのに有効であり、任意に予め定められたフィールド・レングス制限を受けることのない可変性フィールド・レングスの記述を有してなる請求項6記載のデータベース・マネージメント・システム。

8. 上記承継物手段は、パーツのファミリーを明記した最初のユーザのユニットが予め定められたメジャ・ユニットに自動的に変換され、そしてパーツのファミリーを明記した次位のユーザのユニットは他のメジャ・ユニットには自動的に変換されることがないようにされて、ユニット・メジャ変換のためのルール・システムを含んでなる請求項6記載のデータベース・マネージメント・システム。

9. 上記承継物手段は、インスタンスについてのイグジスティング・テキスト

ヤル情報を、上記インスタンスのためシェーマの範囲内でパラメータ値に変形させる手段と、上記インスタンスがシェーマにおいて位置すべきクラスを自動的に評価する手段とを含んでなる請求項6記載のデータベース・マネージメント・システム。

10. 階層性オブジェクト指向シェーマについて、パラメータ・アトリビュート・サーチを実行する手段を更に備えてなる請求項3記載のデータベース・マネージメント・システム。

11. 予め設定されたサーチ基準に密接にマッチするインスタンスと同様に、予め設定されたサーチ基準に正確に応答する総てのインスタンスを復活させるために上記情報のオブジェクト指向表現をサーチする手段を更に備えてなる請求項3記載のデータベース・マネージメント・システム。

12. 予め設定されたサーチ基準の一部分にマッチするインスタンスと同様に、予め設定されたサーチ基準に正確に回答する総てのインスタンスを復活させるために上記情報のオブジェクト指向表現をサーチする手段を更に備えてなる請求項3記載のデータベース・マネージメント・システム。

13. 予め設定されたサーチ基準の一部分にマッチするインスタンスと同様に、予め設定されたサーチ基準に正確に回答する全総てのインスタンスを復活させるために上記情報のオブジェクト指向表現をサーチする手段を更に備えてなる請

求項3記載のデータベース・マネージメント・システム。

14. ダイナミック・クラス・マネージャ、コネクション・マネージャ、クエリ・マネージャ、ハンドル・マネージャ、ユニット・マネージャ、データベース・マネージャ及びファイル・マネージャを有するノリッジ・ベース・サーバと、階層で配列されたオブジェクトとしてインスタンスのクラスを表現しているオブジェクト指向階層シェーマであって、当該シェーマは、階層においてペアレントであるいかなるクラス・オブジェクト、及び階層においてディセendantであるいかなるクラス・オブジェクトについての情報を含んだ各クラス・オブジェクトと密接に結合されており、上記オブジェクトは上記ダイナミック・クラス・マネージャによってマネージされているものと、

クライアント・アプリケーションをオブジェクト指向階層制シエマにアクセスさせるアプリケーション・プログラミング・インタフェースとを備えてなるクライアント/サーバ・アーキテクチャを有するネットワーク。

15. 上記情報のオブジェクト指向階層制シエマにおける1つのクラスは、少なくとも、ハンドルを持ったクラス・オブジェクトとして表現され、このクラス・オブジェクトはそのペARENT・クラスであるとするペARENT・ハンドルを持つと共に、サブクラス・リストを持っており、このサブクラス・リストは上記クラス・オブジェクトのサブクラスであるとする一連のクラス・ハンドルを備え、またこのクラス・オブジェクトはそのアトリビュートであるとするために用いられるハンドルのリストを備えたアトリビュート・リストを含むと共に、サブトリー・インスタンス・カウントを含んでおり、このサブトリー・インスタンス・カウントは上記クラス・オブジェクトに属しかつ上記クラス・オブジェクトの全ディセンダントの中にあるインスタンスの総数の示度数でなる請求項14記載のネットワーク。

16. 上記情報のオブジェクト指向表現における1つのクラスは、少なくともと

、ハンドルを持った次位のクラス・オブジェクトとして表現され、この次位のクラス・オブジェクトはそのペARENT・クラスであるとするペARENT・ハンドルを持つと共に、ローカル・リストを持ち、このローカル・リストは上記次位のクラス・オブジェクトに属するインスタンスであると確認するために用いられ得る一連のハンドルを備え、またこの次位のオブジェクトはそのアトリビュートであるとするために用いられ得るハンドルのリストを備えたアトリビュート・リストを含むと共に、サブトリー・インスタンス・カウントを含んでおり、このサブトリー・インスタンス・カウントは上記次位のクラス・オブジェクトに属しかつ上記クラス・オブジェクトの全ディセンダントの中にあるインスタンスの総数の示度数でなる請求項15記載のネットワーク。

17. 大多数の他のクライアント・アプリケーションが上記オブジェクト指向階層制シエマの次位の部分を動かし又はサーチしている間に、1つのクライ

ント・アプリケーションによって上記オブジェクト指向階層制シェーマの最初の部分の変更を可能とするオブジェクト指向ロック・マネージャを更に備えてなる請求項16記載のネットワーク。

18. プロセッサと、

スクリーンを備え、プロセッサに結合されている表示装置と、

プロセッサに結合されているマウスと、

プロセッサでアクセスされるノリッジ・ベースであって、当該ノリッジ・ベースは、組織化されたものに同じ大多數のパーツのための記述情報を有すると共に、インスタンスのクラスを表現しているパーツ情報についての階層制シェーマを備え、このパーツ情報についての階層制シェーマはルート・クラスを有すると共に、ディセendant・クラスの大多數のレベルを備え、このルート・クラスは大多數が最初のレベルのディセendant・クラスのペアレントであり、この最初のレベルのディセendant・クラスのいくつかは、少なくとも、それぞれの次のレベルのディセendant・クラスの最初のレベルのペアレントであり、この次のレベルのディセendant・クラスのいくつかは、それぞれのその

次の次のレベルのディセendant・クラスの次の次のペアレント・クラスであり、クラスやインスタンスは大多數のアトリビュートを持ち、ここでのパーツ情報についての階層制シェーマにおけるレベル『 n 』のクラスは、階層制シェーマにおけるレベル『 $n-1$ 』のそのペアレント・クラスからアトリビュートを受け継いでいるというものと、

スクリーンのトリー表示エリアにグラフィカル・トリー階層を表示する手段であって、このグラフィカル・トリー階層は、パーツ情報についての階層制シェーマのあまねく選択された一部のクラスを表現しており、上記クラスはスクリーンのトリー表示エリアに表示された個々のクラスのための個別に結び付けられているアイコンを持っているものと、

パーツ情報についての階層制シェーマの一部を表現しているトリー表示エリアの選択された位置でマウスをクリックすることにより、グラフィカル・トリー階層を動かす手段と、

スクリーンのアトリビュート表示エリアにアトリビュートを表示する手段であって、このアトリビュート表示はトリー表示エリアとは異なっており、アトリビュートを表示するこの手段は、アトリビュート表示エリアに表示されたアトリビュートが、パーツ情報についての階層制シェーマにおいてあまねく選択された位置によって類似のアトリビュートであるようなクラスを表示する手段とコーディネートされているというもの

とを備えたパーツ・マネージメント・システム。

19. 上記情報のオブジェクト指向階層制シェーマにおける1つのクラスは、少なくとも、ハンドルを持ったクラス・オブジェクトとして表現され、このクラス・オブジェクトはそのペアレント・クラスであるとするペアレント・ハンドルを持つと共に、サブクラス・リストを持っており、このサブクラス・リストは上記クラス・オブジェクトのサブクラスであるとする一連のクラス・ハンドルを備え、またこのクラス・オブジェクトはそのアトリビュートであるとするために用いられるハンドルのリストを備えたアトリビュート・リストを含むと共に、サブトリー・インスタンス・カウントを含んでおり、このサブトリー・インスタンス

・カウントは上記クラス・オブジェクトに属しかつ上記クラス・オブジェクトの全ディセendantの中にあるインスタンスの総数の示度数でなる請求項18記載のパーツ・マネージメント・システム。

20. 上記情報のオブジェクト指向表現における1つのクラスは、少なくとも、ハンドルを持った次位のクラス・オブジェクトとして表現され、この次位のクラス・オブジェクトはそのペアレント・クラスであるとするペアレント・ハンドルを持つと共に、ローカル・リストを持ち、このローカル・リストは上記次位のクラス・オブジェクトに属するインスタンスであると確認するために用いられ得る一連のハンドルを備え、またこの次位のオブジェクトはそのアトリビュートであるとするために用いられ得るハンドルのリストを備えたアトリビュート・リストを含むと共に、サブトリー・インスタンス・カウントを含んでおり、このサブトリー・インスタンス・カウントは上記次位のクラス・オブジェクトに属しかつ上記クラス・オブジェクトの全ディセendantの中にあるインスタンスの総数の

示度数でなる請求項19記載のパーツ・マネージメント・システム。

21. トリー表示エリアの選択された位置でマウスをクリックしてグラフィカル・トリー階層を動かしているときに、選択されたクラスに属し、かつ上記クラスのディセendantの中にあるインスタンスの総数を表現しているサブトリー・インスタンス・カウントに応じ、表示エリアに見い出されるパーツの数値を表示する手段を更に備えてなる請求項20記載のパーツ・マネージメント・システム。

22. ノリッジ・ベース・クライアントと、

ダイナミック・クラス・マネージャ、コネクション・マネージャ、クエリ・マネージャ、ハンドル・マネージャ、ユニット・マネージャ、データベース・マネージャ及びファイル・マネージャを有するノリッジ・ベース・サーバと、

クラスを含むオブジェクト指向階層制データベース構造であって、ここでの各クラスは、上記クラスのための情報であるとする階層位置を含むクラス・オブジ

ェクト・データ構造で表現されており、このクラス・オブジェクト・データ構造はダイナミック・クラス・マネージャによってマネージされているというものと、

大多数のクライアント・アプリケーションによってアクセスをコントロールするオブジェクト指向ロック・マネージャであって、上記オブジェクト指向ロック・マネージャは、クラス・シェア・ロック、トリー・アップデート・ロック及びトリー・イクスクルージブ・ロックを用いるがインスタンス・ロックを用いることなく、同時にコントロールを行うというものと、

ロック・ホルダ・テーブルであって、このロック・ホルダ・テーブルはロック・マネージャによって使用されており、このロック・マネージャは、要求されるロックがこのロック・ホルダ・テーブルの中にあるイグジスティング・ロックとかわ合わないときに、クライアント・アプリケーションに対し適切なロックを与えることで、上記クライアント・アプリケーションで同時アクセスをコントロールするというもの

とを備えたクライアント／サーバ・アーキテクチャにおけるオブジェクト指

向データベース・マネージメント。

【発明の詳細な説明】

発明の名称

オブジェクト指向データベース管理システム

発明の背景

本発明はデータの高速読み取りのために最適化されるオブジェクト指向データベース管理システムに関し、そして特に、広範な更新が必要とされない応用に適している。項目が値を有していないとき、何も記憶しないように、データベースは構成される。それ故、空の値を記憶することによりメモリ空間が浪費されることなく、また、このような空のフィールドをサーチして、時間が消費することもないので、サーチ速度が改善される。

本発明の種々の特徴は他の応用においても有利に使用することができるけれども、本発明は、製造オペレーションにおける部品及び要素の管理の問題を参照してここでは説明する。本発明は、長期間存続する部品の管理の問題を解決するのに特に有用である。

しばしば、製造会社の競争における成功は主として、マーケットに製品をより早くもたらす会社の能力に依存するかもしれない。目標を達成することのできる企業に対する報酬はかなりのものであるかもしれない。この目標を達成することができないことに対するペナルティは、顧客、或いはマーケット全体の損失であり得る。典型的会社において、部品選択プロセスを再設計することにより、会社のオペレーションをかなり改善し、かつマーケットに製品をより早くもたらす際に大きな利益を得ることができる。さらに、かなりのコスト節約を達成することができる。

設計工学は、コストを減少させ、かつ品質を改善して、マーケットにより早く製品をもたらす競争的原動力の中心であった。会社は、設計技術者をより効率的にするために常に努力している。この効率探求は、設計活動のためのより効果的なツールの提供に変換され、それによって、設計活動はより重要な部分となる。

製造会社における各設計サイクルの頂点は、資材目録を完成させる部品選択プ

ロセスである。設計工学は典型的には、設計要求を満足する数十の部分の規定し

かつ選択しなければならない。あらゆる場合に、設計技術者は、大きな戦略上の影響を会社に与える選択の機会が与えられるであろう。各設計技術者が、各部品を規定しかつ選択するときに面する暗黙の選択は、彼又は彼女が現存の部品を再使用することができるかどうか、又は彼又は彼女が新たな部品を投入する必要があるかどうかの問題である。

この一見簡単な問題への設計技術者の解答に依存して、会社は、新たな部品を相当のコストで提供する必要もあるかもしれない。もしびったり合う部品、又は許容可能な代換部品が既に存在してさえ、大会社の設計技術者は新たな部品を加えることは容易であると考えているということを、マーケットリサーチは示している。なぜなら、部品を調べて、現存する部品が適しているかどうかを判断するためにずっと多くの時間がかかり、面倒であるためである。しかしながら、一つの新たな部品の投入でさえ、時間及び費用がかかる。図1は、新たな部品を投入する前に、製造会社が経験する典型的プロセスを要約している。

現存の認可された部品が設計に使用することができるならば、設計技術者は設計のために多くの時間を有し、新たな部品を投入する費用のかかるプロセスは避けられ、そして従来の部品投入プロセスの価値は最大化されるであろう。しかしながら、必要なものは、使用のために部品を探し出す迅速かつ容易な方法である。部品が迅速に見つけられ、かつ使用されるとき、設計者の利益のみならず、設計工学部門、調達、製造、客先サービス、及び設計技術者下流のあらゆる他のグループは同様に利益を得るであろう。典型的ではあるが、投入部品を探すために多くの価値ある時間を費やし、それを見つけられなかった後、設計技術者は単に別の新たな部品を指定する。

設計者がしばしば部品を見つける困難性を有する理由は、部品を参照する大部分のシステムは、部品番号によりそれをするためである。設計者は、機能的属性、幾何学的記述、及び必要とされるものの他の特性を知っているが、しかし、正しい部品を識別することのできる部品番号をほとんど知らない。過去のこの問題に注意を向ける努力により、部品記述をキーワードを通して利用可能にした。しかし、しばしば、この記述は標準化されず、通常、フィールド長の制限によりむ

き出しの資格記述に制限されている。過去において、投入部品情報への正確なアクセスは、通常不適切であった。

現存の部品を使用する必要性のために、特別の支えが多くの組織により開発されてきた。これらは、インテリジェント部品ナンバリングシステム、クリブシート、資材目録から得られた”使用場所”システム、グループ技術、CADドロー管理システム、そして必用に応じて記述駆動RDBMSアプリケーションを含んでいる。これらの解決策は、以下の理由のためにその場限りのものである。

1. これらの支えは、完全な解決策ではない。それらはしばしば、ジョブをなすために現存部品選択及び投入プロセスの迂回につながる。

2. それらは、他の主要なタスクのために設計されており、かつ一般に役に立たないか、又はこの応用において誤用されるツールに基づいている。

3. この組織は、ビジネスの焦点に直接関係しないリソースを開発し、かつ適用する。

4. あまりに多くの人が、彼らの現在の設計焦点のみの部品選択エキスパートになり、新たなプロジェクトへの人員の機動力を制限する。

5. 適当な部品を見つける頻度を測定するか、或いは新たに投入された部品と使用のために既に利用可能のものとの間で冗長性を測定するいかなるツールもない。

6. 必然的に完全なシステムを開発するこれらの試みは不満足なものであり、かつ捨て去られる。

この問題に取り組むこれらの過去の試みが、なぜ完全な解決策として特徴付けられないのかの付加的な理由がある。それらは、会社の投入部品の全プールを適切に指示しない。この部品プールは一般に、包括的な管理の邪魔をし、かつ現存システムによる完全な訂正行為を抑える特性を有している。このような特性は、データが会社中を横断して、多くの異なるシステムの間で、広くまき散らされる傾向があるという事実を含んでいる。大部分の部品の記述は貧弱であり、かついくつかの部品は記述の矛盾のために、見つけることができない。多くの同様な部品がある。異なっているが、しかしもし部品が識別することができる

ならば、同じ設計基準を満足する部品がある。部品のプールのほとんど小さくならない。一般に、それが今日いかに大きくても、それは明日さらに大きくなるであろう。

過去に、特別の解決策が、キーワードサーチツールを利用することによりこの問題に取り組むために常に試みられてきた。ユーザー特定のキーワードでのサーチは、多くのリレーショナルデータベースアプリケーションの一部である。リレーショナルデータベースでのキーワード照会は、一般にデータベースにあるテキスト文字列に対して特定のテーブルをサーチさせる。これらのアプリケーションは、ワイルドカード、大文字と小文字の区別に対するオプション、或いはキーワード一致と関連した他の機能をサポートすることができる。しかしながら、前述の典型的部品記述に矛盾があると、キーワードアプローチは、それらの有効性が厳しく制限された。キーワードサーチをサポートするデータベースにおいて、問題はキーワードに関して述べられ、そして解答が返されるが、しかし、全ての想定し得る解答が返されるかどうかはわからない。部品管理システムにおいて、データベース内の全ての項目及び全ての関連しているか、或いは同様な項目を見つけることが重要である。さもなければ、適当な部品が会社のデータベース内に以前に存在していないという保証はなく、そしてこのシステムが適当な現存する部品を見つけることができないとき、現存するデータベースに付加されるべき新たな部品を作るコストを、不必要に被るかもしれない。

表1に示された例は、キーワードに基づいた部品検索システムの限界を明らかにしている。表1に示された4つの項目は、部品データベースにおける典型的項目の例を表している。用語“カム従動子ベアリング”での初歩的キーワードサーチは、1つの部品、#0002のみを発見して、多分復帰するであろう。高機能のシステムは、#0001、#0002、#0003において、3つの部分的一致でいくつかの用語を発見して、復帰するかもしれない。“トラックローラー”が“カム従動子”の同義語であるということを知っているシステムは、ありそうにない。また、ローラーとニードルはときには同義語である。“インチ”に関連するサーチは、最初に挙げた部品において一致を見つけるのみであろう。終わりに、最後の2つの部品は、それらがベアリングであるということを意味している

けれども、この記述はそれらをそういうものとしてははっきりとは識別しない。

表 1

部品 #	記述
0 0 0 1	ベアリング、カム従動子、ローラー、1. 0 インチ
0 0 0 2	カム従動子ベアリング、ニードル、1. 0 ”
0 0 0 3	カム従動子、1. 0 ”、ローラー
0 0 0 4	トラックローラー、1. 0 ”

一般的部品記述において、種々の単位に対して異なって規定することは普通ではないので、過去に問題が生じていた。例えば、ある部品は、摂氏の温度で規定された温度特性を有するかもしれないけれども、他のものは華氏で規定されているかもしれない。また、あるネジは、1 インチとして表される長さを有しているかもしれない。別のネジは、2. 5 センチメートルとして表される長さを有しているかもしれない。両方のネジは、同じ設計要求に対して許容できる代換え品であるかもしれない。しかしながら、従来のデータベース管理ツールは、1 インチと 2. 5 センチメートルの両方を包含する長さ範囲を有する現存部品のサーチにตอบสนองして両方の部品を見つけるようには、単位を満足的に扱うことはできなかった。

標準リレーショナルデータベース管理システム (RDBMS) モデルは、部品管理解決策を開発するのに満足的ではない。必然的に内部で開発された会社のシステムが、標準 RDBMS 技術で構築され、かつ一般に、エンドユーザーには満足的ではなかった。

これらの問題を扱う努力において、ある会社は、投入プロセスのドルコスト見積もりを開発し、そしてそれを設計工学グループに戻すメカニズムを提供した。このようなアプローチの背景にある理論的解釈は、設計技術者が結果にどうにか影響を与えることのできる唯一の者であるということである。新たな部品の指定を選択することにより、設計技術者は、図 1 に示されたもののような一連のプロセスステップを会社にゆだね、それは次に、会社に時間及びコストを負わせる。しかしながら、このような努力は満足的ではなく、そして部品を探して、現存部品が適しているかどうかを判断するより迅速かつ容易なシステムの必要性を示す

こととなっている。

可能なときはいつでも新たな部品の投入を避けることにより、図1に示されたプロセスの熟発を完全に取り除くことが望ましいであろう。過去に、設計技術者は、既に投入された部品を指定しかつ選択するために必要なツールを提供しなかった。もし現存部品が設計要求、或いは受容可能な代換え品に完全に一致するならば、以前に部品投入により招いたプロセスコスト（時間及び努力）をダブルするようなことを会社が避けるためのアプローチを提供する必要があった。

会社の現存部品データのプールは、潜在的に価値ある財産であるが、しかし、その効果的な価値は、従来の会社の知識及び投資の有用かつ容易に利用可能なリソースであるデータを禁止する前述の特性によって割り引かれる。それ故、現存部品データのこのプールを有用な情報リソースに手ごろに変換することのできる解決策は、会社にとって大きな価値があるであろう。しかしながら、それを管理する有効なツールは、過去には利用できなかった。

発明の概要

本発明は、その好適具体例において、検索手段、知識ベースクライアント手段、及び知識ベースサーバー手段を含むことができる。本発明と関連して使用するための体系に、好ましくはレガシー手段が含まれて、現存レガシーデータベースの組織化を容易にする。好適具体例において、知識ベースサーバー手段は、ダイナミッククラスマネージャ手段、接続マネージャ手段、質問マネージャ手段、ハンドルマネージャ手段、単位マネージャ手段、データベースマネージャ手段、及びファイルマネージャ手段を含んでいる。好適システムはまた、このシステムへの認知されていないユーザーアクセスを制御するために、登録サーバー手段及びライセンスマネージャ手段を含んでいる。

本発明は、多数の有利な特徴を有する部品管理システムを提供するために使用することができる。本発明のシステムは、設計要求に対する完全な一致又は受容可能な代換え品のいずれかである投入部品を、もしこのような部品が存在するならば、直観的に、決定的に、かつ事実上瞬時に見つけることを可能にする設計技術者のためのツールを提供する。二重化部品は取り除くことができ、かつ同様に在庫品コストも減少する。

オブジェクト指向知識ベースの使用を通して、本発明は、直観的に、瞬時に、決定的に部品データにアクセスすることができ、かつ全ての部品を包含することができる。本発明は、会社の不完全に管理された現存部品データのプールを、価値ある会社の財産に変換することができる。それは、全ての部品情報の仕様、投入、及びその後の検索への継続一貫性及び管理を提供することができる。

部品クラス、サブクラス、形状のような部品特性、材質、及び寸法は、とりわけ、本発明のオブジェクト指向環境に非常によく適合する。部品は、部品ファミリー即ち"スキーマ"内でオブジェクトとして取り扱われる。

本発明は、一般的キーワードサーチよりも決定的利点を提供する属性サーチを使用する。表1を参照して前述したキーワード一致と関連した不完全サーチ問題は、同じデータがパラメトリック属性として再構成されるとき、解決することができる。パラメトリック属性記述は、以下のものから構成される。(1)全ての専門用語をある標準形に縮めること、(2)オブジェクト又はサブクラスに関連した属性の値として各タームを記述すること、及び(3)オブジェクトの属性の組を整理すること。この場合、カム従動子ベアリングは、"取り付けられたベアリング"と呼ばれるベアリングのサブクラスの下に分類される。これは、表2に示されている。このように記述すると、部品は容易に関連づけられ、かつ同じ部品に相当するようになる。これは、キーワードサーチからは明らかにならないであろう。

表 2

部品 #	オブジェクト	取付タイプ	エレメント	直径
			タイプ	
0001	取り付けたベアリング	カム従動子	ローラー	1.0
0002	取り付けたベアリング	カム従動子	ローラー	1.0
0003	取り付けたベアリング	カム従動子	ローラー	1.0
0004	取り付けたベアリング	カム従動子	ローラー	1.0

本発明において、ユーザーは、部品を記述する属性を選択することにより部品データベースをサーチする。選択は、一般から詳細な部品属性にシフトすること

から成る。全ての可能な質問は属性にリンクされ、ユーザーは単に列挙された可能性から選択する。このシフトメカニズムは、望まない部品をマスクする効果を有している。その意図は、サーチ基準に完全に適合する部品を残すことであるが、しかし、適合するかもしれないいかなる部品も取り除くことはない。

本発明は、有効な、継続している部品仕様、記述、及び検索システムである。部品は、それらの関連した属性を使ってそれらを記述することにより見つけられる。属性は、パラメトリック（長さ、キャパシタンス、等）と非パラメトリック（コスト、優先、等）の両方であり得る。記述プロセスは、時折のユーザーにとっては直観的であり、かつ特殊なコンピュータ専門技術を必要としない。必要とされる部品は、事実上瞬時に見つけることができる。この性能レベルは、システムの広範な使用を助長する。応答時間は、任意の時間においてサーチされるデータベース、及びユーザー数とは事実上無関係である。

本発明のシステムは、データへの決定的なアクセスを提供する。もし必要な部品が存在するならば、ユーザーはそれを見つけることができるであろう。もし部品が存在しないならば、ユーザーはそれもまた確実に知り、そのため新たな部品を自信を持って投入することができる。このシステムは、記述基準の下位集合に完全に一致するか又は満足する全ての部品と共に、その記述基準に完全に適合する全ての部品を検索することができる。このシステムは、望まれる属性に基づいた部品の選択を容易にする。望まれる属性の例は、（設計標準化を助長する）”標準”値、（信頼性を保証する）低フィールド故障率、低ユニットコスト、及び優先サプライヤを含んでいる。

本システムは、会社の現存部品レガシーデータのプールを使用可能の情報に手頃に変換することができる。本発明は、設計技術者が、重要な工学属性に基づいた部品の記述を簡単に作成し、かつ編集することを可能にする。全ての部品記述は、部品タイプの関数としての内容及びフォーマットに関して標準化することができる。この記述は、任意、かつ所定のフィールド長制限とは無関係であり、そして異なる部品タイプの種々のフィールド長要求に自動的に適応することができる。このシステムは、内部又は外部の現実により誘引される大きな変化を収容するよう容易に変更することができる点で柔軟性がある。これは、部品ファミリー

全体の追加及び削除、新たな製品ライン、企業整備、併合、買収を含んでいる。

本発明は、単位測定変換可能性を提供する。ユーザーは、彼又は彼女の測定単位を選択して部品を指定することができる。このシステムは、部品の測定単位の変換を支配するルールを提供する。ある部品ファミリーにとって、単位の変換が許容され、かつ必要とされ、他のものにとって、変換は禁止される。このシステムは、どの部品ファミリーにどのルールを適用するかを知っている。

本システムは、いずれかの他の応用又は企業を横断するシステムへの接続性を有するオープンシステム環境を提供する。全ての部品情報にアクセスする企業範囲のデスクトップが提供される。新たに指定された部品に関する部品情報は、瞬時に企業全体で利用可能である。工学技術と部品管理に関係した他の部門との間に情報時間の遅れを無くすことにより、同時並行の工学実践が助成される。このシステムはまた、このシステムへの部品の投入と関連した管理及び制御機能を提供する。

本システムは、設計技術者及び他のユーザーが、パラメトリック及び非パラメトリック属性に関してそれらを記述することにより部品を探し出すことが可能になる。それは、標準及び専用部品の両方に適応するために、部品ファミリー及び属性のダイナミック管理（追加、削除、及び操作）をサポートする。それは、部品仕様及び記述プロセスの管理において継続する構成、一貫性及び制御を提供する。それはまた、継続するシステムにおいて、会社の現存する（レガシーの）部品を含んでいる。

本発明は、知識ベースクライアント及び知識ベースサーバーから成るクライアント/サーバーアーキテクチャーにおいて有利に使用することができる。本発明は、読取り指向のオブジェクト指向データベース管理システムのための特に有利な同時制御メカニズムを提供する。好適具体例において、知識ベースサーバーは、オブジェクト指向ロックマネージャ、ダイナミッククラスマネージャ、接続マネージャ、質問マネージャ、ハンドルマネージャ、ユニットマネージャ、データベースマネージャ、及びファイルマネージャを含んでいる。

本発明のオブジェクト指向ロックマネージャは、多数の有利な特徴を有する同時制御メカニズムを提供するために使用することができる。本発明のシステムは

、設計技術者のためのツールの利用性を最大にする。本発明は、他のユーザーによって変更がなされている間に、クラスオブジェクトを質問しかつ表示することを中断することなく可能にすることにより、最適利用可能性を提供する。これらの変更は好ましくは、クラスの追加、削除、及び編集、属性、インスタンス、及びパラメータを含むであろう。

本発明は、クラスオブジェクトに基づいたロック継承物を使用することにより同時制御システムの性能を最適化する。ロックマネージャ手段は、ロックのサブクラス継承物もなく、ロックをクラスに位置させるメカニズムを実施する。このメカニズムがクラスロックである。ロックマネージャ手段はまた、ロックのための継承メカニズムを提供する。継承メカニズムはツリーロックである。クラスをロックするツリーは、子孫クラスにクラスロックを物理的に位置決めする必要もなく継承物によりそのクラスの全ての子孫をロックするであろう。本発明は、真の共有ロック及び排他的ロックを使用する。本発明はまた、“更新”ロックとして参照される、共有ロックと排他的ロックの間のハイブリッドであるロックモードの新規な実行を提供する。

本発明は、クラスレベルロック粒度を使用することによりロックされるべき必要のある多数のオブジェクトを簡単化することにより性能を最適化する。クラスロックの粒度又は範囲は、クラス自身であり、その属性はクラスによって限定され、そしてそのインスタンスはそのクラスと関連している。本発明はまた、属するクラスとは無関係にロックされるべきインスタンスを許容しない。知識ベースクライアント手段は、オブジェクト指向ロック手段メカニズムを使用して、適切な粒度及び継承物のロックを定めて、これらの手段を使うツールの最大利用可能性、安定性、及び性能を提供する。

本発明のさらなる特徴及び利点は、好適具体例の図面及び以下の詳細な記述と関連して認められるであろう。

図面の簡単な説明

図1は、典型的な通常の部品管理プロセスを示すフローチャートである。

図2は、本発明と関連して使用するのに適している典型的ネットワーク環境の図である。

図3は、本発明のシステムのためのアーキテクチャー全体を示すブロック図である。

図4 A及び4 Bは、このシステムにアクセスするためのログイン手順を示すフローチャートを表している。

図5は部品仕様ウィンドーを示す初期ディスプレイスクリーンを示している。

図6は、サーチの間の部品仕様ウィンドーの例を示している。

図7は、クラスを選択するためのフローチャートである。

図8は、部品カウント及びディスプレイを更新するための手順を示すフローチャートである。

図9は、クラスをオープンするための手順を示すフローチャートである。

図10は、部品仕様ウィンドー内に表示される情報を示すディスプレイスクリーンを示している。

図11は、オープンクラスを閉じるための手順を示すフローチャートである。

図12は、テキストサーチ基準を選択するための手順を示すフローチャートである。

図13は、部品仕様ウィンドー内に表示される情報を示すディスプレイスクリーンを示している。

図14は、数字のサーチ基準を選択するための手順を示すフローチャートである。

図15は、カスタム数字ダイアログボックスを示している。

図16は、部品仕様ウィンドー内に表示される情報を示すディスプレイスクリーンを示している。

図17は、ブールサーチ基準を選択するための手順を示すフローチャートである。

図18は、部品仕様ウィンドー内に表示される情報を示すディスプレイスクリーンを示している。

図19は、列挙したサーチ基準を選択するための手順を示すフローチャートである。

図20は、部品仕様ウィンドー内に表示される情報を示すディスプレイスクリーン

ーンを示している。

図 2 1 は、部品仕様ウィンドー内に表示される情報を示すディスプレイスクリーンを示している。

図 2 2 は、ディスプレイのための属性階数を選択するための手順を示すフローチャートである。

図 2 3 は、サーチ結果を表示するための手順を示すフローチャートである。

図 2 4 は、サーチ結果ウィンドー内に表示される情報を示すディスプレイスクリーンを示している。

図 2 5 は、質問をするための手順を示すフローチャートである。

図 2 6 は、部品情報を表示するための手順を示すフローチャートである。

図 2 7 は、部品情報ウィンドー内に表示された情報を示すディスプレイスクリーンを示している。

図 2 8 は、ユーザー行為を始めるための手順を示すフローチャートである。

図 2 9 は、図 2 8 に示された手順により始められるユーザー行為の一例を示すディスプレイスクリーンを示している。

図 3 0 は、ユーザーが作動ボタンを操作するとき続く手順を示すフローチャートである。

図 3 1 は、部品仕様ウィンドー内に表示される情報を示すディスプレイスクリーンを示している。

図 3 2 は、ユーザーが編集ボタンを作動するとき続く手順を示すフローチャートである。

図 3 3 は、ユーザーがソートボタンを作動するとき続く手順を示すフローチャートである。

図 3 4 は、ソートダイアログボックス内に表示される情報を示すディスプレイスクリーンを示している。

図 3 5 は、ユーザーが部品を編集するとき続く手順を示すフローチャートである。

図 3 6 は、部品エディターウィンドー内に表示される情報を示すディスプレイスクリーンを示している。

図 3 7 は、部品エディターウィンドー内に表示される情報を示すディスプレイスクリーンを示している。

図 3 8 は、ユーザーが部品を削除するとき続く手順を示すフローチャートである。

図 3 9 は、ユーザーが部品を移動するとき続く手順を示すフローチャートである。

図 4 0 は、部品エディターウィンドー内に表示される情報を示すディスプレイスクリーンを示している。

図 4 1 は、クラスのための内部オブジェクト表示を示している。

図 4 2 は、包括的なソストを示している。

図 4 3 は、属性データのためのデータ構造を示している。

図 4 4 は、列挙オブジェクトのためのデータ構造を示している。

図 4 5 は、ユニットファミリーのためのデータ構造を示している。

図 4 6 は、ユニットのためのデータ構造を示している。

図 4 7 は、ユニットファミリーのためのデータ構造を示している。

図 4 8 は、列挙して得られたユニットのためのデータ構造を示している。

図 4 9 は、インスタンス及び関連したパラメータのためのデータ構造を示している。

図 5 0 は、パラメータのためのデータ構造を示している。

図 5 1 は、インスタンスと共にスキーマの一例を示している。

図 5 2 は、ハンドルマネージャがオブジェクトの仮想メモリアドレスの要求に如何に応答するかを示すフローチャートである。

図 5 3 は、ダイナミックファイルのシーケンシャルレイアウトを示している。

図 5 4 は、スキーマ及びインスタンスファイルの一般的レイアウトを示している。

図 5 5 は、ファイルヘッダのレイアウトを示している。

図 5 6 は、知識ベース内のクラスを表すスキーマファイルオブジェクトのレイアウトを示している。

図 5 7 は、知識ベース内の属性を表すスキーマファイルオブジェクトのレイア

ウトを示している。

図 5 8 は、知識ベースにおける列挙を表すスキーマファイルオブジェクトのレイアウトを示している。

図 5 9 は、知識ベースにおけるユニットを表すスキーマファイルオブジェクトのレイアウトを示している。

図 6 0 は、知識ベースにおけるユニットファミリーを表すスキーマファイルオブジェクトのレイアウトを示している。

図 6 1 は、インスタンスファイルオブジェクトのレイアウトを示している。

図 6 2 は、キャラクタ文字列を記憶するために使用されるタイプ 1 ダイナミックオブジェクトのレイアウトを示している。

図 6 3 は、4 バイト長のデータ項目を記憶するために使用されるタイプ 2 ダイナミックオブジェクトのレイアウトを示している。

図 6 4 は、パラメータデータを記憶するために使用されるタイプ 3 ダイナミックオブジェクトのレイアウトを示している。

図 6 5 はスキーマにクラスを如何に付加するかを示すフローチャートである。

図 6 6 は、図 6 5 のフローチャートの続きである。

図 6 7 は、列挙された属性の付加を示すフローチャートである。

図 6 8 は、図 6 7 のフローチャートの続きである。

図 6 9 は、インスタンスの付加を示すフローチャートである。

図 7 0 は、図 6 9 のフローチャートの続きである。

図 7 1 は、クラスの削除を示すフローチャートである。

図 7 2 は、図 7 1 のフローチャートの続きである。

図 7 3 は、属性の削除を示すフローチャートである。

図 7 4 は、図 7 3 のフローチャートの続きである。

図 7 5 は、インスタンスの削除を示すフローチャートである。

図 7 6 はサブツリーの移動と関連したステップを示すフローチャートである。

図 7 7 は、図 7 6 のフローチャートの続きである。

図 7 8 は、最初の親からのクラス移動のホックをはずすのを示すフローチャートである。

図 7 9 は、移動されるべきクラスの共通祖先を見つけるためのプロセスを説明するフローチャートである。

図 8 0 は、図 7 9 のフローチャートの続きである。

図 8 1 は、接続マネージャにより保守されるデータのグラフ表示である。

図 8 2 は、ローカル質問の提示を説明するフローチャートである。

図 8 3 は、図 8 2 のフローチャートの続きである。

図 8 4 は、サブツリー上で質問をするためのプロセスを示すフローチャートである。

図 8 5 は、質問カウントの提示を示すフローチャートである。

図 8 6 は、ロック機能のグラフ表示である。

図 8 7 は、ジェニック (genic) における論理の一致を示している。

図 8 8 は、スキーマエディタウインドー内に表示された情報を示すディスプレイスクリーンを示している。

図 8 9 は、スキーマエディタウインドー内に表示された情報を示すディスプレイスクリーンを示している。

図 9 0 は、クラスツリーのナビゲーションを示すフローチャートである。

図 9 1 は、スキーマエディタウインドー内に表示された情報を示すディスプレイスクリーンを示している。

図 9 2 は、新たなサブクラスにクラスをリペアレントすることを示すフローチャートである。

図 9 3 は、スキーマエディタウインドー内に表示された情報を示すディスプレイスクリーンを示している。

図 9 4 は、スキーマエディタウインドー内に表示された情報を示すディスプレイスクリーンを示している。

図 9 5 は、スキーマエディタにおけるクラスの再配置を示すフローチャートである。

図 9 6 は、クラスマネージャにおけるレガシー手順全体のフローチャートである。

図 9 7 は、スキーマエディタウインドー内に表示された情報を示すディスプレイ

イスクリーンを示している。

図98は、スキーマエディタウインドーにおいて新たなクラスを付加することを示している。

図99は、スキーマエディタウインドー内に表示された情報を示すディスプレイスクリーンを示している。

図100は、スキーマエディタウインドー内に表示された情報を示すディスプレイスクリーンを示している。

図101は、スキーマエディタにおける属性の再配置を示すフローチャートである。

図102は、スキーマエディタウインドー内に表示された情報を示すディスプレイスクリーンを示している。

図103は、スキーマエディタウインドー内に表示された情報を示すディスプレイスクリーンを示している。

図104は、スキーマエディタウインドー内の新たな列挙属性の付加を示すフローチャートである。

図105は、スキーマエディタウインドー内に表示された情報を示すディスプレイスクリーンを示している。

図106は、数字属性の付加を示すフローチャートである。

図107は、スキーマエディタウインドー内に表示された情報を示すディスプレイスクリーンを示している。

図108は、スキーマエディタウインドー内に表示された情報を示すディスプレイスクリーンを示している。

図109は、ブール属性の付加を示すフローチャートである。

図110は、スキーマエディタウインドー内に表示された情報を示すディスプレイスクリーンを示している。

図111は、新たな文字列属性の付加を示すフローチャートである。

図112は、スキーマエディタウインドー内に表示された情報を示すディスプレイスクリーンを示している。

図113は、列挙の付加及び挿入を示すフローチャートである。

図 1 1 4 は、スキーマエディタウィンドー内に表示された情報を示すディスプレイスクリーンを示している。

図 1 1 5 は、スキーマエディタウィンドー内に表示された情報を示すディスプレイスクリーンを示している。

図 1 1 6 は、列挙タイプ属性の削除を示すフローチャートである。

図 1 1 7 は、スキーマエディタウィンドー内に表示された情報を示すディスプレイスクリーンを示している。

図 1 1 8 は、スキーマエディタ内の数字属性を編集するためのフローチャートを示している。

図 1 1 9 は、スキーマエディタウィンドー内に表示された情報を示すディスプレイスクリーンを示している。

図 1 2 0 は、テーブルへの値の付加を示すフローチャートである。

図 1 2 1 は、スキーマエディタ内のテーブルエディタにおける自動値ダイアログのピクチャーである。

図 1 2 2 は、テーブルエディタ内にラベルを付加するためのプロセスのフローチャートである。

図 1 2 3 は、スキーマエディタ内のテーブルエディタにおける自動ラベリングダイアログのピクチャーである。

図 1 2 4 は、テーブルの行及び列をユーザーが変更するためのプロセスフローチャートを表している。

図 1 2 5 は、インポートのためのコマンドラインパラメータを示している。

図 1 2 6 は、シンプのためのコマンドラインパラメータを示している。

図 1 2 7 は、スキーマエディタにおける属性をユーザーが削除するためのフローチャートである。

図 1 2 8 は、クラスがローカルに限定された属性を有していないとき編集する属性のために選択されたクラスを持つスキーマエディタ内のスクリーンのピクチャーである。

図 1 2 9 は、クラスが編集のために利用可能の属性を有しているとき編集する属性のために選択されたクラスを持つスキーマエディタ内のスクリーンの別のピ

クチャーである。

図 1 3 0 は、属性を削除するときスキーマエディタに現れる確認ダイアログを示している。

図 1 3 1 は、ジェニック内の一致基準の一例を示している。

図 1 3 2 は、レガシー処理のためのプロセスを示すフローチャートである。

図 1 3 3 は、レガシーマネージャの自動部品分類機能を示すフローチャートである。

図 1 3 4 は、レガシープロセスにおける部品を分類するための方法を示すフローチャートである。

図 1 3 5 は、祖先の部品のレガシーを示すフローチャートである。

図 1 3 6 は、インスタンスをレガシーするための方法を示すフローチャートである。

図 1 3 7 は、図 1 3 6 のフローチャートの続きである。

図 1 3 8 は、図 1 3 7 のフローチャートの続きである。

図 1 3 9 は、分類のためにクラスの属性の処理を示すフローチャートである。

図 1 4 0 は、スキーマオブジェクトのための分類語彙の処理を示すフローチャートである。

図 1 4 1 は、クラス分類語彙のレガシーを示すフローチャートである。

図 1 4 2 は、部品インスタンスのパラメータ化を示すフローチャートである。

図 1 4 3 は、クラスのための非数値属性のレガシーを示すフローチャートである。

図 1 4 4 は、ソート要求を処理する前、及び後の質問結果のステートを示す図である。

図 1 4 5 は、数値属性のためのレガシー内部プロセスを示すフローチャートである。

図 1 4 6 はクラスファイヤーの内部アクションを示すフローチャートである。

図 1 4 7 は、図 1 4 6 のフローチャートの続きである。

図 1 4 8 は、図 1 4 7 のフローチャートの続きである。

図 1 4 9 は、図 1 4 8 のフローチャートの続きである。

図150はスキーマ発生器の内部アクションを示すフローチャートである。
図151は、図150の続きである。
図152は、ダイナミッククラスマネージャにおけるデータベースマネージャ内のデータ構造を示している。
図153は、インポートの内部プロセスのフローチャートである。
図154は、図153の続きである。
図155は質問を提示した後の質問マネージャ内のデータ構造を示している。
図156は、数字質問セクターをセットした後の質問マネージャ内のデータ構造を示している。
図157は、ブール質問セクターをセットした後の質問マネージャ内のデータ構造を示している。
図158は、質問マネージャ内の数字質問セクタークラスを示している。
図159は、質問マネージャにおける列挙した質問セクター及び文字列質問セクタークラスを示している。
図160は、質問マネージャにおけるベース質問又はクラス及びブールセクタークラスを示している。
図161は、質問マネージャにおける質問結果クラスを示している。
図162は、質問マネージャにおけるベース質問クラス、質問クラス、及びサーチ結果クラスを示している。
図163は、質問マネージャにおけるメインデータ構造である質問マネージャクラス及び質問ハンドルマネージャクラスを表している。
図164は、質問が作成された後の質問マネージャにおいて作成されるクラスを示している。
図165はジェニクにおける処理のステージを示すフローチャートである。
図166は、図165のフローチャートの続きである。
図167は、図166のフローチャートの続きである。
図168は、本発明の典型的サーバーアーキテクチャーを示している。
図169は、本発明の典型的クライアントアーキテクチャーを示している。
図170は、レガシー知識ベースオープンスクリーンのためのプロセスフロー

チャートを示している。

図 1 7 1 は、レガシー知識ベースオープンダイアログを示している。

図 1 7 2 は、レガシーアプリケーションが起動されるとき現れるスクリーンを示している。

図 1 7 3 は、レガシーワークエリアが選択された後のプロセスのためのフローチャートである。

図 1 7 4 は、メインレガシースクリーンである。

図 1 7 5 は分類語彙編集のためのクラスを選択後のスクリーンを示している。

図 1 7 6 は、分類語彙編集のためのダイアログを起動するフローチャートである。

図 1 7 7 は、クラスのための分類語彙編集ダイアログを示している。

図 1 7 8 は、図 1 7 7 の分類語彙編集ダイアログのためのプロセスフローを示している。

図 1 7 9 は新たな項目を付加した後の分類語彙編集ダイアログを示している。

図 1 8 0 は、分類語彙項目に入力されるテキストを示している。

図 1 8 1 は、分類語彙項目内の通常の表示を示している。

図 1 8 2 は、新たな分類語彙項目を挿入した結果を示している。

図 1 8 3 は、分類語彙項目における複雑な通常の表示を示している。

図 1 8 4 は、列挙した属性のための分類語彙エディタを起動するためのフローチャートを示している。

図 1 8 5 は、部品仕様ウィンドーからの列挙属性のための分類語彙エディタを持ち出すための手順を示すディスプレイスクリーンを示している。

図 1 8 6 は、部品仕様ウィンドーからの列挙分類語彙の編集を示すディスプレイスクリーンを示している。

図 1 8 7 は、編集部品ウィンドーからの列挙分類語彙の編集を示すディスプレイスクリーンを示している。

図 1 8 8 は、ソートされた質問結果からインスタンスを検索する要求を処理する前及び後の質問結果のステートを示す図である。

図 1 8 9 は、ソートされた質問結果内のソート範囲の管理を示している。

図190は、編集部品ウィンドーからの数字属性分類語彙エディタを持ち出す手順を示すディスプレイスクリーンを示している。

図191は、編集部品ウィンドーからの数字属性分類語彙を編集するための手順を示すディスプレイスクリーンを示している。

図192は、ユニット分類語彙を編集するための手順を示すディスプレイスクリーンを示している。

図193はユニット分類語彙を編集するためのフローチャートを示している。

図194は、選択された部品のためのレガシー処理をセットアップするための手順を示すディスプレイスクリーンを示している。

図195は、選択された部品のためのレガシー処理をセットアップするためのフローチャートを示している。

図196は、レガシー選択部品の結果を示すディスプレイスクリーンを示している。

図197は、パラメータ化するために属性のリストを編集するためのフローチャートを示している。

図198は、パラメータ化するために属性のリストを編集するための手順を示すディスプレイスクリーンを示している。

図199は、カスタマーデータからカスタマースキーマを発生するためのフローチャートを示している。

図200は、最初にカスタマーデータを分類し、かつインポートマップを発生するためのフローチャートを示している。

図201は、ベンダー部品のデータベースからカスタマーデータを補強するためのフローチャートを示している。

図202は、ネットワーク性能を最適化するために質問結果をバッファするためのフローチャートを示している。

図203は、非列挙の分類語彙の編集を示している。

図204は、本発明の好適具体例に適しているネットワーク環境の図である。

図205は、本発明の好適具体例を使用するシステムのための全アーキテクチャーを示すブロック図である。

図 2 0 6 A は、過去に提案された拡張データベース粒度ヒエラルキーを示す図である。

図 2 0 6 B は、過去に提案されたロックグラニューールのヒエラルキーの別の例を示す図である。

図 2 0 6 C は、本発明のロックグラニューールのヒエラルキーを示す図である。

図 2 0 7 A は、クラス共有ロックが 3 つのクラスに適用されたヒエラルキーを示す図である。

図 2 0 7 B は、ツリーロックがクラスに適用され、かつ図 2 0 7 A と関連して、ロックサブサミングの一例を示すヒエラルキーを示す図である。

図 2 0 8 は、本発明により使用されるロックタイプ及び粒度のためのロック衝突を表す図である。

図 2 0 9 は、ロック要求を許可するプロセスにおいて 1 ステップ中のヒエラルキーを示す図である。

図 2 1 0 は、ロック要求を許可するプロセスにおいて次のステップの間のヒエラルキーを示す図である。

図 2 1 1 は、図 2 0 9 及び図 2 1 0 に示されたステップが首尾よくいったときのクラスでツリーロック要求を許可するプロセスにおいて次のステップの間のヒエラルキーを示す図である。

図 2 1 2 は、検索ウィンドーがオープンされるときの実行されるロックプロセスを表すフロー図である。

図 2 1 3 は、クラスがクラスヒエラルキー内で選択されるときの生じるプロセスを示している。

図 2 1 4 は、サブクラスを表示するためにクラスをオープンするプロセスを表すフロー図である。

図 2 1 5 は、ユーザーが“クラス発見”活動を選択するとき生じるプロセスを表すフロー図である。

図 2 1 6 は、クラスをオープンしかつ選択することによりスキーマをナビゲートするときのスクリーンディスプレイの一例を示している。

図 2 1 7 は、図 2 1 6 のディスプレイに相当するスキーマ内のクラスの内部ロ

ツクステートの一例を示すスキーマの図である。

図 2 1 8 は、図 2 1 7 に示されたスキーマと相互関連しているようなロックマネージャにより保守されるロックテーブルを示している。

図 2 1 9 は、図 2 1 8 に示されたロックテーブル内のロックオブジェクトの 1 つの内容を示す図である。

図 2 2 0 は、ユーザーが知識ベース内のクラスに部品を付加するとき生じるプロセスを示す図である。

図 2 2 1 は、部品が付加されているクラスを有するスキーマを示している。

図 2 2 2 は、図 2 2 0 に示されるような部品を付加するプロセスのためのロックテーブルセットを示している。

図 2 2 3 は、図 2 2 1 - 2 2 2 に相当する付加部品オペレーションのためのクラスに相当するロックオブジェクトを示している。

図 2 2 4 は、スキーマに部品を付加するときのスクリーンディスプレイの一例を示している。

図 2 2 5 は、ユーザーが編集部品機能を選択した一例のフローチャートを示している。

図 2 2 6 は、ユーザーが編集部品ウィンドー内にある間に、クラスヒエラルキーツリー内の異なる位置にナビゲートする一例のフローチャートを示している。

図 2 2 7 は、部品を編集するときのスクリーンディスプレイの一例を示している。

図 2 2 8 は、図 2 2 7 において編集されているスキーマに相当する 1 スキーマを示している。

図 2 2 9 は、編集部品ウィンドーの作成の完了後のロックホルダーテーブルを示している。

図 2 3 0 は、図 2 2 7 - 2 2 9 に示された例に相当するロックオブジェクトを示している。

図 2 3 1 は、サブツリー内の 1 クラスから所定のサブツリー内の別のクラスに単一の部品を移動する一例のフローチャートを示している。

図 2 3 2 は、サブツリー内の 1 クラスからそのサブツリー内の別のクラスに任

意数の部品を移動する一般の場合の一例のフローチャートを示している。

図 2 3 3 は、サブツリー内の 1 クラスからそのサブツリー内の別のクラスに任意数の部品を移動する一般の場合のプロセスの間のロックホルダーテーブルを示している。

図 2 3 4 は、図 2 3 2 に示された部品移動の一般の場合のソース及び宛先クラスのためのロックオブジェクトの詳細、及び関連したアクションを示している。

図 2 3 5 は、部品移動オペレーションと関連した好適ディスプレイを示している。

図 2 3 6 は、1 つの部品が知識ベースから取り除かれるべきである最適化ケースのプロセスを示すフローチャートである。

図 2 3 7 は、サブツリーから 1 以上の部品を削除する一般の場合のプロセスを示すフローチャートである。

図 2 3 8 は、クラスから 1 インスタンスを取り除くことを望むロックホルダーにより保持されなければならないロックを示している。

図 2 3 9 及び 2 4 0 は、部品削除オペレーションと関連した好適ディスプレイを示している。

図 2 4 1 は、スキーマの構造を変更するためにスキーマエディタを使用するとき一致制御に関係しているステップを記述するフローチャートである。

図 2 4 2 は、図 2 4 1 に記述されたオペレーションの間に保持されるロックを示すロックテーブルを示している。

図 2 4 3 は、図 2 4 1 に示されたプロセスの 1 ステップ内にオープンされているスキーマディベロップメントウィンドーを示す好適具体例のためのスクリーンディスプレイを示している。

図 2 4 4 は、インスタンスを表示するとき一致制御手段により使用されるメカニズムを例示するフローチャートを示している。

図 2 4 5 は、図 2 4 4 に示された状況に対するロックホルダーテーブルの条件を示す、ロックテーブル、スキーマの図、及びロックオブジェクトの 1 つに関する細部を示している。

図 2 4 6 は、図 2 4 4 に示されたプロセスの 1 ステップ内にオープンされる

ーチ結果ウィンドーを示す好適具体例のためのスクリーンディスプレイを例示している。

図247は、スキーマ編集をするための承認を要求するためのステップを示すフローチャートである。

図248は、インスタンス編集をするための承認を要求するためのステップを示すフローチャートである。

図249は、クラス共有ロックを要求するためのステップを示すフローチャートである。

図250は、ツリー共有ロックを要求するためのステップを示すフローチャートである。

図251は、ツリー更新ロックを要求するためのステップを示すフローチャートである。

図252は、ツリー排他的ロックを要求するためのステップを示すフローチャートである。

図253は、知識ベースクライアントによるロックマネージャの適用を表すチャートである。

図254は、ロックマネージャにより使用されるロックテーブルの図である。

図255は、ロックホルダーテーブルのためのデータ構造を示している。

図256は、ロックホルダー開始のオペレーションを示すフローチャートである。

図257は、ロックホルダー終了のオペレーションのためのフローチャートである。

図258は、知識ベースサーバーのためのコンピュータハードウェア構成の主要な構成要素を示している。

図259は、検索、スキーマエディタ、グラフィカルユーザーインターフェース成分、及びAPIのためのコンピュータハードウェア構成の主要な構成要素を示している。

図260及び図261は、部品属性を比較するプロセスのためのフローチャートを示している。

図 2 6 2 A は、サーチ結果ウィンドーのディスプレイの一例を示している。

図 2 6 2 B は、比較部品ダイアログボックスのディスプレイの一例を示している。

図 2 6 3 は、選択部品比較コマンドが起動した後、比較部品ダイアログボックスのディスプレイの一例を示している。

図 2 6 4 は、部品仕様ウィンドーを示す初期ディスプレイスクリーンを示している。

図 2 6 5 は、サーチ中の部品仕様ウィンドーの一例を示している。

図 2 6 6 は、部品仕様ウィンドー内に表示された情報を示すディスプレイスクリーンを示している。

図 2 6 7 は、ユーザーが部品を編集するとき続く手順を示すフローチャートである。

図 2 6 8 は、部品エディタウィンドー内に表示される情報を示すディスプレイスクリーンを示している。

図 2 6 9 は、部品エディタウィンドー内に表示される情報を示すディスプレイスクリーンを示している。

図 2 7 0 は、ユーザーが部品を削除するとき続く手順を示すフローチャートである。

図 2 7 1 は、ユーザーが部品を移動するとき続く手順を示すフローチャートである。

図 2 7 2 は、部品エディタウィンドー内に表示される情報を示すディスプレイスクリーンを示している。

図 2 7 3 は、クラスのための内部オブジェクト表示を示している。

図 2 7 4 は、包括リストを示している。

図 2 7 5 は、属性データのためのデータ構造を例示している。

図 2 7 6 は、列挙オブジェクトのためのデータ構造を例示している。

図 2 7 7 は、インスタンス及び関連したパラメータのためのデータ構造を示している。

図 2 7 8 は、パラメータのためのデータ構造を示している。

図 279 は、インスタンスを持つスキーマの一例である。

図 280 は、スキーマにクラスを如何に付加するかを示すフローチャートである。

図 281 は、図 280 におけるフローチャートの続きである。

図 282 は、列挙属性の付加を示すフローチャートである。

図 283 は、図 282 のフローチャートの続きである。

図 284 は、インスタンスの追加を示すフローチャートである。

図 285 は、図 284 のフローチャートの続きである。

図 286 は、クラスの削除を示すフローチャートである。

図 287 は、図 286 のフローチャートの続きである。

図 288 は、属性の削除を示すフローチャートである。

図 289 は、図 288 のフローチャートの続きである。

図 290 は、インスタンスの削除を示すフローチャートである。

図 291 は、サブツリーの移動と関連したステップを示すフローチャートである。

図 292 は、図 291 のフローチャートの続きである。

図 293 は、最初の親からのクラス移動のアンフックを示すフローチャートである。

図 294 は、移動されるべきクラスの共通祖先を見つけるためのプロセスを示すフローチャートである。

図 295 は、図 294 のフローチャートの続きである。

好適具体例の詳細な説明

本発明は、ネットワーク環境で好都合に使用することができる。多数の構成が可能であるが、ここでは、1つの例のみを説明する。本説明は、例示的であり、かつ本発明はここで説明した特別の例或いは構成に限定されないということが理解できよう。適切なネットワーク環境の概観が図 2 に示されている。ネットワーク 100 は、第一の UNIX サーバーホスト 101 を包含している。1 以上の知識ベースが第一の UNIX サーバーホスト 101 上にインストールされる。例示した例において、第一の知識ベースサーバーデーモン 102 が第一の UNIX サ

ーバーホスト101上で実行される。データは、時には、二次記憶装置として参照される第一のディスクドライブ103上に物理的に記憶することができる。1以上の知識ベースサーバー102がこのシステム100上に備えることができる。例えば、第二の知識ベースサーバーデーモン104を備えることができる。同様に、データは第二のディスクドライブ105上に物理的に記憶することができる。第一のUNIXサーバーホスト101は、第二のUNIXサーバーホスト106及び第三のUNIXサーバーホスト107とネットワーク上で通信することができる。この例において、登録サーバーデーモン108は、第二のUNIXサーバーホスト106上にインストールされる。登録サーバーデーモン108は、知識ベースサーバーデーモン102及び104と同じUNIXサーバーホスト101上で実行することができるであろう。登録サーバー108によって使用される情報を包含するあるファイルは、第三のディスクドライブ109上に物理的に記憶することができる。登録サーバー108は、ユーザーの特徴へのアクセス及び知識ベースへのアクセスを管理するために使用される。登録サーバー108はまた、システムアドミニストレータにより、異なるユーザープロフィールを異なるタイプのユーザーのためにセットアップする。例えば、知識ベース123から部品を検索するアクセス許可のみを必要とするユーザーがいるかもしれない。他のユーザーは、部品を付加し、或いは現存部品を編集するアクセス許可を必要とするかもしれない。登録サーバー108は、ユーザーが特定の機能にアクセスするのを規定しかつ制御する便利な方法を提供する。登録サーバー108は、使用中の知識ベース、このシステムの使用が許可されているユーザー、及び各ユーザーがリストにした知識ベースに対して有するアクセス権を記述している。

ライセンスマネージャサーバーデーモン110が、第三のUNIXサーバーホスト107上にインストールされる。ライセンスマネージャサーバー110は、ネットワーク100上で認知されたユーザーに利用可能なライセンス数を管理する。ライセンスマネージャ110は、“フローティング”ライセンスを使用する。例えば、20ライセンスがライセンスマネージャ110を通して利用可能の時、ネットワークのどの20ユーザーも、これらのライセンスを同時に使用することができる。

知識ベースサーバー 102 がスタートすることができる前に、ライセンスマネージャサーバー 110 及び登録サーバー 108 は、作動していなければならない。登録サーバーデーモン 108 が作動を継続するために、ライセンスマネージャサーバー 110 からライセンスを得ることができなければならない。もし登録サーバー 108 がライセンスマネージャサーバー 110 にコンタクトすることができないならば、それは、終了する。それ故、第一にライセンスマネージャサーバー 110 が開始されるであろう。第二に、登録サーバー 108 が開始されるであろう。その後、知識ベースサーバー 102 が開始されるであろう。

ユーザーは、ネットワーク 100 に接続された適当なワークステーション 111 を使って、知識ベースサーバーデーモン 102 又は知識ベースサーバーデーモン 104 を通して利用可能のデータにアクセスすることができる。例えば、好ましくは、X11R5/Motif v1.2 ソフトウェアを動作させるサンマイクロシステムズ SPARC ステーション 111 である。或いは、SPARC 互換のワークステーションを使用することができる。さらに、X11R5 を持つ Motif v1.2 を動作させるヒューレットパッカードシリーズ 700 ワークステーションがまた満足的な結果を与えるであろう。例示された例において、サンマイクロシステムズ SPARC ステーション 111 は、SunOS 4.1.x 動作システムを作動する。ヒューレットパッカードシリーズ 700 プラットフォームは、好ましくは HP-UX 9.0.x ソフトウェアを含んでいる。

さらに、ユーザーは、マイクロソフトウインドーズ v3.1 を作動する IBM PC 互換コンピュータ 112 を使ってネットワーク 100 にアクセスすることができる。例示された例において、IBM PC 互換コンピュータ 112 は、386、486 或いはペンティアムベースのマシンにすることができる。IBM PC 互換コンピュータ 112 は、ディスプレイ 113、マウス 114、及びキーボード 115 を含んでいる。ディスプレイ 113 は、好ましくは VGA 又は SVGA CRT 113 である。例示された例において、IBM PC 互換コンピュータ 112 は、MS-DOS 5.0 又はそれ以降のディスクオペレーティングシステム、好ましくは MS-DOS 6.2 を作動している。IBM PC 互換コンピュータ 112 はまた、ウインドウズ 1.1 対応の TCP/IP ソフトウェアを有してい

なければならない。IBM PC互換コンピュータ112を使うウインドウズクライアントは、知識ベースサーバー102と通信するためにTCP/IPを通してRPCコールを使用するであろう。IBM PC互換コンピュータ112は、ソフトウェアインストールのために十分利用可能のディスクスペースを有するべきである。例示された例において、IBM PC互換コンピュータ112はまた、少なくとも4メガバイトのRAMメモリ、好ましくは16メガバイトのメモリを有するべきである。

サンマイクロシステムズSPARCステーション111は同様に、ディスプレイ116、マウス117、及びキーボード122を有している。

図2に例示されたネットワーク100はまた、コンピュータ118を使用するXウインドウズクライアントをサポートし、かつこれは、ディスプレイ119、マウス120、及びキーボード121を有している。ユーザーは、ワークステーション111とインターフェースする適切なエミュレーションモードでXウインドウズを使ってこのシステムにアクセスすることができる。

図2に示された例において、サーバーホスト101、106、及び107のそれぞれが、サンマイクロシステムズSPARCステーション（又はSPARC互換）、或いはヒューレットパッカードシリーズ700コンピュータにすることができる。目下好適の具体例において、ネットワーク上で単一のUNIXシステムが、知識ベースサーバーデーモン102、登録サーバーデーモン108、及びライセンスマネージャサーバーデーモン110を起動させるために指定することができる。この実施は、管理を容易にすることができる。最良の性能のために、本発明を具体化するソフトウェア及び知識ベースは、単一のサーバーホスト101のローカルディスクドライブ103上に常駐するであろう。しかしながら、例えば知識ベース123は、リモートディスクドライブ109上に常駐することができる。

この例において、ネットワーク環境は、ファイルシステムを持つオペレーティングシステムを含み、仮想メモリをサポートし、UDP/TCP/IPプロトコルを使用し、そしてONC/RPC（オープンネットワークコンピューティング/リモートプロシージャコール）サービスを提供する。さらに、もしネットワー

ク環境がマルチプロセス及びマルチタスクをサポートするならば、有用である。

本システムは、ユーザーによる対話型編集をサポートする。ユーザーは部品属性を付加しかつ削除することにより部品スキーマを変更することができ、かつこれらのカスタム部品をサポートするためにスキーマにセクション全体を付加することができる。スキーマ編集に加えて、データベース内の部品は、修正し、追加し、かつ削除すると同時に、ヒエラルキーにスキーマ内に再位置決めすることができる。

本発明は、オブジェクト指向ツールセットを提供し、かつこれは、(1) ダイナミッククラス管理をサポートし、(2) 多数の部品(例えば、数十万を越える部品)を有するデータベースをサポートし、(3) 数百人のユーザーによる部品の対話型検索をサポートするのに十分な性能を有し、(4) 異なる測定単位間での変換を理解し、かつ自動的に管理する。このシステムは、知識ベース管理システムとして参照することができる。

本知識ベース管理システムは、ユーザーが“オブジェクト”を、それらの属性に関してそれらを記述することにより探し出すことができる。部品管理アプリケーションにおいて、これらのオブジェクトは、部品であるが、しかし、それは、属性の集合により記述されるいかなる他の項目であるかもしれない。アプリケーションは、素材専門家(コンピュータプログラマーではない)により作成される。洗練されたアプリケーションは、コンピュータプログラム開発ではなく、素材ベースのスキーマの開発と結びついている。

本発明は、目下好適の具体例の全アーキテクチャーの以下の説明と関連してより良く理解することができる。

1. 全アーキテクチャー

さて、図3を参照すると、目下好適の具体例は、リトリーバー130、知識ベースクライアント131、及び知識ベースサーバー132を備えることができる。レガシーマネージャ133は、好ましくは、本発明と関連して使用するためのスキーマ内に現存のレガシーデータベースを組織化するのが容易にするために備えられる。好適具体例において、知識ベースサーバー132は、ダイナミックラスマネージャ134、接続マネージャ135、質問マネージャ136、ハンド

ルマネージャ 137、ユニットマネージャ 138、データベースマネージャ 139、及びファイルマネージャ 140を含んでいる。好適システムはまた、このシステムへの認知されたユーザーアクセスを管理するために登録サーバー 141及びライセンスマネージャ 142を含んでいる。

スキーマエディタ 144は好ましくは、スキーマを修正し、或いはカスタマイズするために備えられている。アプリケーションプログラミングインターフェース、即ちAPI 143がまた、例示された環境において備えられている。

知識ベース 123は、情報を包含するデータベースであり、かつディスクドライブ 103上に記憶されている。本例における知識ベース 123は、3つのファイル：スキーマファイル、可変データファイル、及びインスタンスファイルから構成される。スキーマは、クラス、属性、ユニット、及びユニットファミリー、及びそれらの関係の集合である。

本例において、実行可能な知識ベースサーバー 132は、pmxdbcである。各pmxdbcサーバーは、1つの知識ベース 123にアクセスを提供する。それ故、UNIXサーバーホスト 101は、各知識ベース 123に対して1つのpmxdbcプロセスを実行しなければならない。例えば、3つの知識ベースを有するシステムにおいて、UNIX ps コマンドは、3つのpmxdbcサーバーが動作していることを示すであろう。

RDBMSベースのアプリケーションとは異なり、本知識ベース管理システムによると、解決策、複雑性、及び応答時間は、サイズ及び関係した数によって指数関数的には増加しない。知識は、ソフトウェアコードの量とは結びついていない。スキーマは、アプリケーションを再コンパイルすることなく動的に更新することができる。データ及びスキーマは、ユーザーが対話的に変更することができる。質問は、相当するインデックスを見つけることに等しく、RDBMSテーブル接合を計算することではない。データベースサイズは、減少する。本発明の知識ベース管理システムデータベース 123は典型的には、等価なRDBMSデータベースのサイズの約1/10である。

本システムにログインするステップは、図4A及び図4Bに示されたフローチャートに示されている。

ログイン手順は、図4 Aのステップ150に示されるように、リトリーバー130にユーザーがログインすることにより開始される。ユーザーの名前及びパスワードは、151に示されるように、登録サーバー141に送られる。ステップ152において、ユーザー名及びパスワードは、登録サーバー141によって有効かどうかチェックされる。もしユーザー名及びパスワードが有効でないならば、フローはステップ150に戻り、そしてユーザーは再び試みなければならない。もし名前及びパスワードが有効であるならば、フローはステップ153に続き、そこで、リトリーバー130は、ライセンスマネージャ142から適切なソフトウェアライセンスを求める。

ステップ154において、ライセンスマネージャ142は、ライセンスがそのユーザーのために利用可能であるかどうかを判断する。もしライセンスが利用可能でないならば、フローは図4 Aに示されるステップ150に戻る。もしライセンスが利用可能であるならば、ライセンスマネージャ142は、動作のためのライセンスを許可し、かつフローはステップ155に続く。リトリーバー130は、利用可能な知識ベース123のリストをディスプレイ116上に表示するであろう。知識ベースのリストは、登録サーバー141から得られる。登録サーバー141はまた、ユーザーがアクセス権を有している知識ベースのリストに戻るのみであろう。ステップ156において、ユーザーはそれから、オープンすべき知識ベース123を選択することができる。

ステップ157において、リトリーバー130は、オープン知識ベース要求を知識ベースサーバー132に送るであろう。ステップ158において、知識ベースサーバーは、要求された知識ベース123がロックされているかどうかをチェックする。例えば、入力アドミニストレータが知識ベース123に広範な入力を実行するとき、知識ベース123をロックすることが望まれるとき、そして、いずれか他のユーザーがそれをアクセスするの一次的に妨げるときがある。知識ベース123がロックされるインスタンスは、典型的にはある人が知識ベース123に非他のアクセスする必要があるものである。もし知識ベース123がロックされるならば、フローはステップ155に戻り、そこで、リトリーバー130は、ユーザーがアクセス権を有する登録サーバー141から知識ベースのリストを

ディスプレイ 116 上に再び表示する。ユーザーはまた、ユーザーが最初にオープンすることを試みた知識ベース 123 がロックされているということをユーザーに通知するメッセージを受信するであろう。

もし要求された知識ベース 123 がロックされないならば、フローは図 4B のステップ 159 に続き、かつ知識ベースサーバー 132 は、オープンモードがこのユーザーまたは知識ベース 123 のために有効であるかどうかを判断するためにチェックする。例えば、もし知識ベース 123 が読取りのみであり、かつユーザーが、知識ベース 123 への書き込みオペレーションが要求されるモードでそれをアクセスすることを試みたならば、フローは図 4A のステップ 155 に戻り、かつユーザーはディスプレイ 116 上にメッセージを受信する。

この例において、もしこの要求されたオープンモードが特別のユーザーのためにその特別の知識ベース 123 を利用可能であるならば、フローは、図 4B に示されたステップ 160 に続く。知識ベースサーバー 132 は、ライセンスマネージャ 142 から適切なソフトウェアライセンスを得るを試みる。もしライセンスが許可されないならば、フローは、図 4A に示されたステップ 155 に戻る。もしライセンスが利用可能であるならば、フローは図 4B に示されたステップ 161 に続く。その場合、知識ベースサーバー 132 は、リトリーバー 130 への接続及び知識ベースハンドルに戻るであろう。ユーザーはそれから、ネットワーク 100 に成功的にログオンし、そして要求された知識ベースサーバー 102 へのアクセスを有するであろう。

図 168 は、知識ベースサーバー 132 のための計算及び通信環境を提供するコンピュータハードウェア構成 101 の主構成要素を示している。この構成は、主メモリ 2101 からプログラムインストラクションをフェッチしかつ実行する演算論理ユニット 2100 を含む中央処理ユニット即ち CPU 2109 から構成される。このプログラムは、ディスクドライブ 103 上に記憶され、そしてそのアクセスは、ディスクコントローラ 2106 を通じて提供される。知識ベースファイル 123 はまた、ディスクドライブ 103 上に記憶され、かつ主メモリ 2101 内の仮想メモリアドレス 2112 を通じてアクセスされ、そして、それを通して、必要なとき、ディスクファイル 2108 内の隣接データのページ 2111

は、主メモリ2101内にコピーされる。本発明の好適具体例は、この知識ベース管理システムのために仮想メモリ2112を使用する。知識ベースサーバ132は、クライアントAPI143と、ローカルエリアネットワーク100を通して相互作用し、かつそのアクセスは、ネットワークコントローラ2102によって、又はワイドエリアネットワーク2104を通して制御され、そしてそのアクセスは、シリアルインターフェースコントローラ2103によって制御される。I/Oバス2105は、CPU2109と、周辺データ記憶装置、インターフェース及び通信構成要素との間のデータ転送を仲介する。

図169は、リトリバー130、スキーマエディタ144、レガシーのグラフィカルユーザーインターフェース要素133、及びAPI143のための計算及び通信環境を提供するコンピュータハードウェア構成の主構成要素を示している。この構成は、主メモリ2101からプログラムインストラクションをフェッチしかつ実行する演算論理ユニット2100を含む中央処理ユニット即ちCPU2109から構成される。このプログラムは、1以上のディスクドライブ2110上に記憶され、かつそのアクセスは、ディスクコントローラ2106を通して提供される。ユーザーは、キーボード115、及びマウス又は同様なグラフィカルポインタ114を通して、CRTディスプレイ113上に表示されるグラフィカルユーザーインターフェースによってこのシステムと相互作用する。API143は知識ベースサーバ132と、ローカルエリアネットワーク100を通して通信し、かつそのアクセスは、ネットワークコントローラ2102により、或いはワイドエリアネットワーク2104を通して制御され、そしてそのアクセスは、シリアルインターフェースコントローラ2103により制御される。I/Oバス2105は、CPU2109と、周辺データ記憶装置、インターフェース及び通信構成要素との間のデータ転送を仲介する。

リトリバー

リトリバー130は、部品を見つけかつ管理するためのグラフィカルインターフェースを提供するアプリケーションである。リトリバー130は、API143を使って知識ベースクライアント131と通信する。リトリバー130は、オブジェクト指向グラフィカルユーザーインターフェースを提供する。ユー

ザーは、キーボード 115 及びマウス 114 を通して入力を提供するリトリバー 130 と相互作用する。リトリバーはディスプレイ 116 上に情報を表示する。

図 5 は、ユーザーがこのシステムに成功的にログオンした後、ディスプレイ 116 のスクリーン上に現れる典型的表示を示している。ここで述べる特別の例は、ウィンドーズ環境で説明するけれども、ウィンドーズにおける実施に限定されないということが理解されよう。当業者は、マウス 114 によって如何に、クリックし、ダブルクリックし、ドラッグし、ポイントし、そして選択するかを含むウィンドーズ技術及びインストラクションに通じている。追加の情報は、マイクロソフトコーポレーション、ワンマイクロソフトウェイ、レッドモンド、ワシントン州、98052-6399、パートナンバー 21669 から利用可能なマイクロソフトウィンドーズユーザーズガイド (1992) から得ることができる。

ユーザーが最初に知識ベース 123 をオープンするとき、図 5 に示されるように、部品仕様ウィンドー 170 が現れる。最初に、スクリーン 171 の左側部分に、発見された部品 172 を表示し、それは、このインスタンスにおいて、知識ベース 123 で発見された部品の総数である。また、スクリーン 171 の左側部分に表示されるのは、ルートクラス 173 及びルートサブクラス 174 である。例示された例において、ルートサブクラス 174 は、電気的要素、機械的 (即ち、機械的要素)、及び材質である。ルートクラス 173 は、親を持たない最上部クラスである。この例において、それは、知識ベース 123 の名前であり、或いはスキーマの全くの初めである。サブクラス 174 は、親を有するクラスである。1 つのクラスが選択されるとき、そのクラスに属する任意のサブクラスが、ディスプレイ 171 上に現れるであろう。サブクラスは、親の子供である。例えば、機械的サブクラス 174 の親は、ルートクラス 173 であり、かつ機械的サブクラス 174 は、親ルートクラス 173 の子供である。図 5 に示された例において、3 つのサブクラスがある。

スクリーン 175 の右側部分は、ルート属性 176 を表示する。例示された例において、属性は、部品番号、記述、及びコストである。属性 176 は、クラス又はサブクラス 174 の特性である。

特定の属性値をサーチ基準177として定めるために入力することができる。コマンドボタン178は、部品仕様ウィンドー170内に表示される。コマンド名が薄暗く表示されるとき、現時点でそのコマンドは利用できない。図5に示された例において、ディスプレイボタン179は、作動するとき、サーチにおけるその点で現在の仕様に一致する部品のリストを表示するであろう。編集ボタン180及びメークボタン181がまた、図5に示されている。これらのコマンドボタンは、ユーザーが属性値を編集することについて認知されており、かつ新たな部品を作るアクセス権を有しているときのみ示される。作動するとき、編集ボタン180は、ユーザーに属性値を編集させる部品エディタウィンドーを表示させる。メークボタン181が作動するとき、それは、ユーザーに、知識ベース123に部品を付加させる。これらの3つのボタンは、コマンドボタン178の部品エリア186に現れる。ディスプレイ指令エリア187は、“全てセット”コマンドボタン182及びクリアコマンドボタン183を有している。全てセットボタン182が作動するとき、それは、各属性にシーケンシャル表示指令を送る。この指令は、サーチ結果ウィンドーにおける属性表示を配置するために使用される。クリアボタン183が作動するとき、それは、全ての属性から表示指令番号を取り除かせる。

クリア基準エリア188は、“全て”ボタン184及び“選択”ボタン185を含んでいる。“全て”ボタン184が作動するとき、それは、全ての値をサーチ基準フィールド177からクリアさせる。“選択”ボタン185が作動するとき、それは、選択されたサーチ基準フィールド177から値をクリアさせる。

スクリーン171の左側部分は、分割バー189によりスクリーン175の右側部分から分離される。ユーザーは、分割バー189を左又は右にドラッグして、スクリーン171の左側部分及びスクリーン175の右側部分のサイズを変更することができる。

アイコンが、ユーザーに情報を提供するために部品仕様ウィンドー170内に表示される。閉じたフォルダーアイコンは、サブクラスを有するクラスを表すために使用される。オープンしたフォルダーアイコン190は、オープンしたクラス173及びサブクラスを表すために使用される。保護されたアイコン191は

、ユーザーが部品を作るとき値を入力することができない属性を示している。未定義のアイコン 192 は、もしユーザーによって選択されるならば、選択された属性 176 のためのいかなる値も有さない部品をサーチするために使用される列を示している。また、図 5 に示されているのは、“部品番号”、“記述”、及び“コスト”と表題を付けた属性のそれぞれと関連したテキストアイコン 193 である。テキストアイコン 193 は、キャラクタの文字列から値を有する属性 176 を示すために使用される。例えば、書き込まれた記述は、テキスト属性である部品の属性 176 である。指令コラム 194 は、そのシーケンスを、左から右に示すために使用され、そしてそこで、属性 176 は、サーチ結果ウィンドー 299 が表示されるとき現れるであろう。適用できるとき、この列において“1”は、サーチ結果ウィンドーの最左に表示される属性 176 を示し、この列における“2”は、次に表示される属性 176 を示し、そして左から右に同様に示す。

部品仕様ウィンドー 170 はまた、質問タイプインディケータ 195 を包含している。これは、部品を編集するためのアクセス権を持つユーザーのためにのみ現れる。これは、ユーザーが実行している質問のタイプ、即ちそれがグローバルであるか、又はローカルであるかを示す。

ユーザーが部品を探し出す必要があるとき、ユーザーは一般にその部品の特性又は属性 176 を知るが、しかしユーザーはその部品番号を知らないかもしれない。その属性 176 を知ることによって、ユーザーは容易に知識ベースにおける部品を探し出すことができる。ユーザーは、ユーザーが見つけることを望む部品のタイプを規定することにより、知識ベースにおける部品を探し出す。ユーザーは、部品のクラス 173 及びサブクラス 174 を選択することにより、そして属性サーチ基準 177 を入力することにより、部品を規定する。

ユーザーが探し出すことを望む部品を規定する際の第一のステップは、部品が属するクラス 173 をオープンすることである。ユーザーが図 5 に示された機械的クラス 174 のようなクラスをオープンするとき、ユーザーは、ヒエラルキーの次のレベル、即ち図 6 に示されたサブクラス 196 を見る。図 5 に示された機械的サブクラスの次の閉じたフォルダーアイコン 189 は、図 6 に示されたオープンフォルダアイコン 190 により置き換えられる。部品を規定する次のステッ

ブは、部品が属する次のサブクラス196、この例においてはファスナーサブクラス196をオープンすることである。ユーザーがサブクラスフォルダー196の1つをオープンするとき、ユーザーは、図6に示されるように、サブクラス197の次のレベルを見る。ユーザーは、図6に示されるボルトサブクラス197のようなサブクラス197の別のレベルをオープンすることにより、同様にサブクラス198及び199の下位レベルを通して、ユーザーが、リーフクラス201と呼ばれる、これ以上のサブクラスを持たないクラスに達するまで、部品を規定し続ける。リーフクラスは、ページアイコン202によって識別される。オープンクラス199は、クラス199のサブクラス204を接続するサブツリーを形成するライン232によって表示されるであろう。特に、ライン232は、クラス199のためのオープンフォルダーアイコン190を、サブクラス204の閉じたフォルダアイコン189と接続する。ライン232は、水平ブランチが垂直ライン232をサブクラス204のための閉じたフォルダアイコン189と接続するように、オープンフォルダアイコン198から最後のサブクラス204のレベルに下方に垂直に延びる。

各サブクラスレベルで、そのクラスで見つけられる部品の数は、見つけられた部品172として表示される。発見部品数172は、サブクラス及びリーフクラス（図6参照）を含む現サブクラス199内に探し出される部品数を示している。ユーザーへのこの瞬時フィードバックは、サーチを大いに容易にする。

リトリバー130に付随するステップは、図7に示されている。ユーザーは、ステップ205におけるクラスを選択する。ステップ206において、選択されたクラスは、強調表示で表示されている（図5参照）。リトリバー130は、ステップ206において現質問を選択されたクラスにリセットする。

図7を参照すると、フローのステップ207において、リトリバー130は、受け継いだ属性176がセットされた質問セクタ177を有しているかどうかを決定する。もし質問セクタがセットされているならば、フローはステップ208に進み、そして、リトリバー130は、受け継いだ属性176のための相当する質問セクタをセットする。それから、フローはステップ209に進む。ステップ207において、もし受け継いだ属性が何らのセットされた質問セク

クタも有していないならば、フローはステップ209に直接進む。

ステップ209において、リトリバー130は、クラス199のためのローカル属性203を得て、そして、それらをディスプレイに、属性ウィンドー175として参照される部品仕様ウィンドー170の右側部分175において付加する。

フローはステップ210に進み、そこで、リトリバー130は、部品カウントを更新し、かつその情報を発見された部品172として表示する。それからフローはステップ211に進み、そこで、制御はユーザーに戻り、そしてシステムは別のコマンドを待つ。

部品カウント及び表示を更新するための手続き210は、図8に詳細に示されている。部品カウント及びディスプレイ更新するために、リトリバー130は質問結果カウントを表す値を得、かつそれを発見部品172として表示する。これは、図8のステップ212に示されている。

それから、リトリバー130は、ステップ213で実行されるが、部品カウントがゼロであるかどうかを判断するためにチェックする。もし部品カウントがゼロであるならば、リトリバー130は、この特別のユーザーが新たな部品を知識ベースに付加するためのアクセス権を有しているかどうかを判断するためにチェックする。これは、ステップ214で生じる。もしユーザーがこのようなアクセス権を有していないならば、フローはステップ211に進み、そしてユーザーに制御を戻す。もしユーザーがこのようなアクセス権を有しているならば、それからリトリバー130は、メークボタン181を作動する。この時点まで、その手続は利用可能でないので、メークボタン181は、薄暗く表示される。好適具体例において、ユーザーが、彼または彼女にそうすることを許可するアクセス権を有していなければ、ユーザーは知識ベースに新たな部品を付加することが許されない。メークボタンの作動は、図8に示されたフローチャートのステップ215で生じる。

もし部品カウントがゼロに等しくないならば、フローはステップ216に移り、そこで、リトリバー130は、ユーザーが部品を編集するアクセス権を有しているかどうかを判断するためにチェックする。もしユーザーがこのようなア

セス権を有しているならば、リトリーバー130はフローのステップ217に進み、そして編集ボタン180を作動する。それから、フローはステップ218に進み、そこで、リトリーバー130は、表示ボタン179を作動する。ステップ216において、もしユーザーが部品を編集するためのアクセス権を有していないならば、フローは直接ステップ218に進む。表示ボタンがステップ218で作動した後、フローはステップ211に進み、そこで、制御はユーザーに戻る。

図9は、クラスをオープンするためにリトリーバー130によって実行されるステップを示している。図6に示されたファスナークラス196のようなクラスをオープンするために、そのクラスのすぐ次の閉じたフォルダ189をポイントするようカーソルを位置決めし、そしてダブルクリックする。図9に示されるように、リトリーバー130は、それから、閉じたフォルダ189の表示を変更し、そしてそれをステップ220においてオープンフォルダアイコン190と置き換える。リトリーバーは、サブクラス197のリストを得る。

ステップ221において、リトリーバー130はサブクラスのリストを通して進み、そしてこのリストにおける次のサブクラスがリーフクラス201であるかどうかを決定する。もしそうならば、フローは図9におけるステップ222に進み、そしてページアイコン202がそのサブクラス201のために表示される。それから、制御はステップ224に進むであろう。

ステップ221において、もしこのリストの次のサブクラスがリーフクラスでないならば、フローはステップ223に進み、そこで、リトリーバー130は、閉じたフォルダアイコン189及びサブクラス197のためのクラス名を表示する。それから、フローはステップ224に進む。

ステップ224において、リトリーバー130は、ディスプレイへのリスト内にこれ以上のサブクラス197があるかどうかを判断するためにチェックする。もしあるならば、フローはステップ221に戻る。もしないならば、フローはステップ205に進む。

クラス199（図10参照）を閉じかつオープンするためにリトリーバー130に付随する手続は、図11に示されている。ステップ225において、ユーザ

ーは、閉じられるべきであるクラス 199 と関連したオープンフォルダアイコン 190 をダブルクリックする。それから、フローは、図 11 のステップ 226 に進む。ステップ 226 において、リトリーバー 130 は、ディスプレイ 171 からサブツリー 232 のための全てのラインを取り除く。サブクラス 204 の名前及びそれらと関連した閉じたフォルダアイコン 189 はまた、ディスプレイ 171 から取り除かれるであろう。サブクラス 204 により以前に占められたスペースを取り除くためにツリー構造の表示がそれから戻されるであろう。

それから、フローはステップ 227 に進み、そこで、親クラス 199 の次のオープンフォルダ 190 は、閉じたフォルダアイコン 189 と置き換えられる。それから、フローは図 11 のステップ 228 に進み、そして制御はユーザーに戻される。

ユーザーがオープンクラスを閉じるとき生じる部品仕様ウィンドウ 170 に対する変更は、図 10 を図 6 と比較することによりより良く認められるかもしれない。図 10 で始めると、もしユーザーが数値クラス 199 を閉じるならば、サブツリーライン 232 及びサブクラス 204 は、ディスプレイから取り除かれ、そして数値クラス 199 と関連したオープンフォルダアイコン 190 は、閉じたフォルダアイコン 189 と置き換えられるであろう。ディスプレイ 171 は、図 6 に示されるように、更新されるであろう。

一組の予め定義された値から取られた値を有するかもしれない属性 203 は、列挙属性として参照される。例えば、ボルトのためのヘッドスタイルは可能なヘッドスタイルの有限なリストから取られた 1 タイプであるのみであるので、ボルトの下のヘッドスタイル属性 203 は、一組の予め定義された値から取られる。ヘッドスタイル 203 のような列挙属性は、関連した列挙アイコン 233 によって識別される。

真又は偽のいずれかの値を持つ属性 203 は、ブール属性である。例えば、ボルトは、それが、自己ロックボルトであるかどうかを示すために使用される属性を有することができる。もしこのボルトが自己ロックでないならば、この属性値は偽である。ブール属性は、それらと関連したブール属性アイコン 234 を有している。

数値である値を有し、かつ関連した測定単位を有する属性236は、数値属性と呼ばれる。例えば、ボルト236の長さは数値属性である。数値属性236は、それらと関連した数値属性アイコン235を有している。

ユーザーはさらに属性サーチ基準177を入力することにより部品を規定することができる。各クラス及びサブクラスは関連した組の属性176を有している。属性176は、部品が作られる材質、その長さ、或いは仕上げのような部品の特性である。ユーザーが追加のサブクラス196、197、198及び199をオープンするとき、これらのサブクラス196、197、198及び199と特に関連した属性203が現れる。これらの属性203は、オープンクラス/サブクラス199のローカル属性であり、かつ受け継いだ属性である存続属性176に追加される。属性サーチ基準177を入力することにより、ユーザーは、ユーザーが適用可能性をチェックする必要がある部品数を限定することができる。

図12を参照すると、選択されたサブクラス又はリーフクラス240に多くの部品があり得る。そして、サーチ基準177を入力することによりさらに部品を規定することが望ましい。ユーザーがサーチ基準177が入力するとき、リトリバー130は直ちに、規定された属性値176、203の全てを有する部品のみを探し出すサーチを実行する。4つのタイプの属性がある。(1)列挙、(2)数値、(3)テキスト、及び(4)ブール。リトリバー130は、各タイプの属性のためのサーチ基準を入力しかつクリアするため異なる手続を有している。

テキスト属性サーチ基準177を入力するための手続が、図12に示されている。ステップ250において、ユーザーはテキスト属性を選択する。例えば、ユーザーは部品番号属性241のためのサーチ基準242を入力することができるであろう。そうするために、ユーザーは部品番号属性241と関連したテキスト属性アイコン193をクリックするであろう。それから、リトリバー130は、図13に示されるように、テキストサーチ基準ダイアログボックス237をポップアップする。リトリバー130は、カーソルを、テキストサーチ基準ダイアログボックス237のデータエントリフィールド243の最左位置に位置決める。図12を参照すると、リトリバー130はそれから、データエントリフィールド243内に入力されるテキスト入力を受け入れるために、ステップ25

1に進む。

リトリバー130は、サーチ基準242の一部として特殊なキャラクタの使用を可能にする。アステリスクは、任意数のキャラクタに一致する。例えば、（ピコファラッドのための）省略形“p f”を含む全ての部品を、選択された部品番号テキスト属性241におけるどこかに定めることが望まれるかもしれない。これを達成するために、ユーザーは、テキストデータエントリフィールド243において* p f *をタイプすることができるであろう。図13に例示された例において、データエントリフィールド243において015*とタイプすることは、いかに多くの追加のキャラクタ又は番号が部品番号内に付随するかに関わらず、数字015で始まるいかなる部品番号も、リトリバー130にサーチさせるであろう。疑問符号は、いずれか一つのキャラクタに一致する。例えば、分散サイズ、即ち、1/8、1/4、1/2、等を含む記述を持つ全ての部品を探し出すために、ユーザーは、?/?とタイプすることができるであろう。特殊なキャラクタ*又は特殊なキャラクタ?をテキスト内に包含する部品をサーチすることが望まれる場合が生じるかもしれない。いずれかの特殊なキャラクタ*又は?の直前に|をタイプすることにより、リトリバー130は、後続の特殊キャラクタを正規のキャラクタとして認識するであろう。

テキスト属性サーチ基準242は、大文字と小文字を区別しない。サーチは、それが大文字であるか或いは小文字であるかにかかわらずキャラクタに一致するであろう。テキストデータエントリフィールド243においてユーザーによりタイプされた文字の大文字又は小文字は、サーチが一致する属性値を探しているとき無視される。

ユーザーは、テキストデータエントリフィールド243におけるユーザーの入力が確認されるまで、制御権を有している。ユーザーは、テキストサーチ基準ダイアログボックス237においてOKボタン244をクリックすることにより、或いはキーボード上115上のエンターキーを押すことにより、このような入力を確認することができる。

このことは、ユーザーが特別のボルトを規定しかつ探し出すためにオープンすることができるクラス及びサブクラスの例の以下の説明と関連してより良く理解

することができる。

テキストサーチ基準ダイアログボックス 237 内にキャンセルボタン 245 が備えられて、ユーザーがテキストサーチを中断するのを可能にする。もしキャンセルボタン 245 が作動すると、リトリバー 130 は、ユーザーが関連したテキスト属性アイコン 193 の選択をした直前の状態に復帰する。テキストデータエントリフィールド 243 内へのいかなる入力も無視されるであろう。

クリアボタン 246 が、テキストサーチ基準ダイアログボックス 237 内に備えられる。もしこのボタンが作動するならば、リトリバー 130 は、テキストデータエントリフィールド 243 内のいかなる入力もクリアするであろう。ダイアログボックス 237 はそのまま留まり、かつリトリバー 130 は、ユーザーからの入力の確認を待ち続けるであろう。

もしユーザーがテキストデータエントリフィールド 243 内にキャラクタを入力し、かつ OK ボタン 244 を作動するならば、フローは図 12 に示されたステップ 253 に続く。リトリバー 130 は、もし関連した属性 241 のために現在何も存在しなかったならば、現在の質問にテキストセクタを付加するであろう。もし予め存在するテキストセクタがあるならば、リトリバー 130 は、現在の質問においてそれを置き換えるであろう。それから、このテキスト属性サーチ基準 242 を含むサーチが実行されて、リトリバー 130 は、図 12 に示されたステップ 210 に進むであろう。

エントリフィールド 243 において、ユーザーにより入力されたテキストデータは、部品使用ウィンドー 170 の右側部分 175 のサーチ基準フィールド 242 において表示されるであろう。部品カウント 172 は、サーチの結果を反映するために図 13 に示されるように更新されるであろう。

もしユーザーによって入力されるテキストデータの長さがデータエントリフィールド 243 のサイズを超えるならば、スクロールボタン 247 が、視野 243 の外側にあるテキストを表示するために当業者に公知の方法で使うことができる。

図 13 に示された長さ属性 236 のような数字属性は、サーチ基準 177 として選択することができる。図 14 を参照すると、ユーザーは、関連した数字属性

アイコン 235 をクリックすることにより、図 13 に示された数字属性長さ 236 のような数字属性を選択する。図 14 を参照すると、これは、フローチャートにおいてステップ 255 により表されている。

それから、リトリバー 130 は、標準値のテーブルが選択された数字属性 236 のために定義されたかどうかを判断するためにステップ 256 に進む。もし標準値のテーブルが定義されなかったならば、リトリバー 130 はステップ 257 に進む。図 15 に示されたカスタム数字ダイアログボックス 265 が現れるであろう。カスタム数字ダイアログボックス 265 は、ある範囲の数値の入力を可能にする。“from” 数字入力フィールド 266 が、カスタム数字ダイアログボックス 265 内に備えられる。“to” 数字入力フィールド 267 がまた、カスタム数字ダイアログボックス 265 内に備えられる。ユーザーが “from” 値をタイプするとき、それは、自動的に “to” 値フィールド 267 にコピーされる。もしユーザーがデフォルト測定単位 268 を使って唯一の値をサーチすることを望むならば、ユーザーは、OK コマンドボタン 270 をクリックすることにより、“from” 入力フィールド 266 と “to” 入力フィールド 267 の両方に同じ値を有する入力を確認することができ、そして、図 14 に示されたステップ 260 に進む。

ステップ 257 において、ユーザーは、デフォルト測定単位 268 以外の異なる測定単位を選択することができる。デフォルト測定単位は、作動するとき他の利用可能な測定単位のリストを包含するドロップダウンリストボックスを生じるボタン 269 と共に、カスタム数字ダイアログボックス内に表示される。このように、異なる測定単位が、ドロップダウンリストボックスから選択することができる。

カスタム数字ダイアログボックス 265 は、当業者に公知の方法で動作するキャンセルボタン 271 及びクリアボタン 272 を含んでいる。カスタム数字ダイアログボックス 265 はまた、前述の OK ボタン 270 を含んでいる。ユーザーの入力は、OK ボタン 270 が強調表示されているときユーザーが OK ボタン 270 をクリックするか、或いはキーボード 115 上のエンターキーを押すとき図 14 に示されたステップ 260 で確認される。

図14を参照すると、ステップ256において、もし標準値のテーブルが数字属性236に対して定義されているならば、ユーザーはステップ258において標準値のテーブル273（図16参照）が与えられる。図16を参照すると、もし数字属性236のために定義されている標準値のリストが、標準値のテーブルのためのポップアップウィンドー273に表示することのできる番号を超えるならば、スクロールボタン274は、標準値ウィンドー273のテーブルにおいて現在表示されていない値を表示するために、当業者に公知の方法でを使用することができる。数字属性236のために定義されている複数の標準値275が、図16に示されるように、標準値ウィンドー273のテーブルに表示される。現在選択された標準値276は、強調して表示される。

標準値ウィンドー273のテーブルは、OKボタン277を含んでいる。リトリバー130は、ユーザーがOKボタン277を動作するとき強調標準値276を受け入れるであろう。標準値ウィンドー273のテーブルはまた、当業者に公知の方法で動作するキャンセルボタン278及びクリアボタン280を含んでいる。

標準値ウィンドー273のテーブルは、カスタムボタン279を含んでいる。もしユーザーが、必要とされる数値と同じである標準値275をリスト上に発見しないならば、ユーザーはカスタムボタン279を動作させることができる。図14において、リトリバー130は、ステップ259においてカスタムボタンの動作をチェックする。もしカスタムボタンが動作すると、フローはステップ259からステップ257に進む。それから、ユーザーは、図15に示されたカスタム数字ダイアログボックス265が与えられる。それから、ユーザーは、図14のステップ257を参照して前述した方法で所望の数値を入力することができる。

ステップ260において、ユーザーがOKボタン277を動作することにより、或いはキーボード115上のエンターキーを押すことにより入力を確認するとき、フローはステップ260からステップ261に進む。リトリバー130は、数字属性236が以前の質問内に存在したかどうかによって、現在の質問内に数字セレクトを付加し、或いは置き換えるであろう。それから、リトリバー

130は、ステップ210に進み、部品カウント172を更新し、そしてディスプレイ170を更新する。標準値テーブルウィンドー273は、ユーザーの入力が、例えば、OKボタン277を作動することにより確認されるとき消えるであろう。ステップ262において、リトリバー130はそれからユーザーに制御を戻し、かつ別のコマンドを待つ。選択された数値は、数字サーチ基準フィールド281内に表示されるであろう。

ブル属性203のためのサーチ基準を入力するための手続が、図17に示されている。ユーザーは、ブル属性アイコン234をクリックすることによりサーチ基準を使うブル属性203を選択する。

図17を参照する。リトリバー130はステップ301に進む。ブルダイアログボックス283がポップアップして、ユーザーに真及び偽の選択を提示する。図18に示されたブルダイアログボックス283は、真オプションボタン284及び偽オプションボタン285を含んでいる。ウィンドウ動作環境において典型的であるように、これらのオプションボタンは、相互に排他的である。ユーザーは、真オプションボタン284又は偽オプションボタン285をそれぞれクリックすることにより、真又は偽サーチ基準282のいずれかを選択することができる。

ブルダイアログボックス283は、OKボタン286を含んでいる。ユーザーは、OKボタン286を作動することにより図17に示されたステップ302において入力を確認する。

ブルダイアログボックス283はまた、キャンセルボタン287及びクリアボタン288を含み、かつこれらの機能は、当業者には公知である。

ユーザーがOKボタン286を作動することにより所望のブルサーチ基準282を確認するとき、リトリバー130は、図17においてステップ303に進み、そして現質問において適切なブルセクタを付加するか、或いは取り替える。前述したように、ブルセクタは、もしこのようなセクタが以前の質問に存在しなかったならば付加されるであろう。ブルセクタは、もしブルセクタが以前の質問においてこの属性203のために現れたならば、置き換えられるであろう。ブルダイアログボックス283が消え、そして選択されたサ

ーチ基準が、部品仕様ウィンドー 170 の右側部分 175 の適切なサーチ基準ワールド 282 において表示される。発見部品 172 の表示はまた、更新されるであろう。

図 19 を参照すると、ユーザーは、列挙属性アイコン 233 をクリックすることにより列挙属性 289 を選択することができる。これは、図 19 に示されたステップ 305 において達成される。それから、リトリバー 130 は、図 20 に示されるように、列挙属性ダイアログボックス 291 を表示する。列挙属性ダイアログボックス 291 は、特別の属性 289 が有することのある有効値のリストを提示する。選択された値が、強調エントリ 293 として表示される。ユーザーは、サーチ基準 290 のための多数値 292 を選択することができる。多数値 292 が選択されるとき、リトリバー 130 は、それらを、サーチ基準 290 のための "or" ロジック条件としてそれらを扱う。言い換えると、サーチは、選択される列挙値 292 のいずれか 1 つを有する部品を検索するであろう。

ダイアログボックス 291 は、クリアボタン 296 を含んでいる。ユーザーは、選択された値 293 の全てを選択解除し、かつ最初からやり直すことができる。これは、多数値 292 が選択されたとき便利である。ユーザーは、ダイアログボックス 291 により提供されるキャンセルボタン 295 を作動することによりこの動作を中断することができる。これにより、図 19 に示されるように、フローは、ステップ 307 からステップ 309 に移る。

ユーザーが選択された列挙値 293 に満足するとき、ユーザーは、ダイアログボックス 291 において提供される OK ボタン 294 を作動することにより入力を確認することができる。図 19 を参照すると、フローは、これが生じるときステップ 307 からステップ 308 に進む。リトリバー 130 は、この属性のために予め存在する値が以前の質問で選択されたかどうかに依存して、現サーチ質問において選択された列挙値 292 を付加し、或いは取り替えるであろう。リトリバー 130 は、それから図 19 に示されるステップ 210 に進み、それからディスプレイを更新するであろう。列挙した属性ダイアログボックス 291 は消えるであろう。選択されたサーチ基準 293 は、サーチ基準ディスプレイワールド 290 に現れる。それから、リトリバー 130 は、図 19 に示されたステ

トップ309に進み、かつ別のコマンドを待つユーザーに制御を戻す。

図20を参照すると、列挙属性ダイアログボックス291はまた、当業者に公知の方法で動作するスクロールボタン297を含んでいる。

図21は、部品仕様ウィンドー170の更新された表示を示している。例示された例において、テキスト属性サーチ基準242は、部品番号属性241と関連したサーチ基準ディスプレイフィールドに表示される。列挙属性サーチ基準29は、列挙属性“ヘッドスタイル”289と関連したサーチ基準フィールドに表示される。数字サーチ基準281が、長さの数字属性236と関連したサーチ基準ディスプレイフィールドに表示される。更新された発見部品表示172は、所望の属性を有する単一の部品を探し出すことにサーチが成功したと云うことを明らかにしている。

サーチ結果を表示すべき階数を決定するために、リトリバー130により使用される手続のフローチャートは、図22に示されている。この手続は、表示結果がサーチの際に探し出される部品の属性に従いソートされるべき階数を選択することがユーザーに許されるステップ310によって始まる。この表示は、サーチ基準として選択されなかった属性176と共に、選択された属性241、289、及び236に従いソートすることができる。選択は、所望の属性241に相当するために階数コラム194のボタン298をクリックすることにより達成される。クリックされる第一の階数ボタン298は、どのサーチ結果が表示のためにソートされるかに関して、第一の属性であろう。“1”が、第一にクリックされる階数ボタン298の表示上に現れる。同様に、クリックされる第二の階数ボタン299は、サーチ結果がソートされる第二の基準であり、そして“2”がボタン299の表示上に現れる。

ユーザーが図21に示されるように階数ボタン298をクリックするとき、リトリバー130は、図22に示されたフローチャートのステップ311に進む。リトリバー130は、要求された階数ボタン298が既にセットされているかどうか、即ち、階数ボタン298上に既に番号を有しているかどうかを判断するためにチェックする。もしそうでないならば、フローはステップ312に進み、そしてリトリバー130は、階数ボタン298を、現在セットされている最

高階数プラス1にセットする。即ち、もしユーザーが図21に示された階数ボタン357をクリックしたならば、そのコスト属性358のための表示階数は、現在セットされていない。この例において、現在セットされていない最高階数は“6”である。このように、ステップ312において、階数ボタン357は、現在セットされた最高階数よりも1大きいので、“7”にセットされるであろう（即ち、 $6+1=7$ ）。それから、“7”が階数ボタン357上に表示される。それから、フローはステップ315に進む。再び、図22に示されたステップ311を参照すると、もし表示階数が特別の階数ボタン299のために現在セットされているならば、フローはステップ313に進み、かつ選択された表示階数は解除される。言い換えると、表示ボタン299は、オフに切り換えられる。それから、フローはステップ314に進み、そして、オフに切り換えられた階数ボタン299の数よりも大きな数を持つ現在セットされている階数ボタン361、358、359、及び360のそれぞれが、1だけデクリメントされ、そしてリセットされるであろう。図21に例示された例において、もしユーザーが（現在“2”を表示する）階数ボタン299をクリックするならば、その階数ボタン299は、リセットされるであろう（それは、ブランク状階数ボタン357が現れるであろう）。さらに、階数ボタン299よりも大きな表示階数に現在セットされた（即ち、“2”より大きい）残りの階数ボタン358、359、360及び361は、1だけデクリメントされるであろう。階数ボタン358は“3”から“2”に変更され、階数ボタン359は、“4”から“3”に、そして、以下同様に変更される。階数ボタン298は、その表示階数が“1”であり、かつリセットされた階数ボタン299の表示階数よりも大きくないので、変更されないであろう。それから、フローは、ステップ315に進み、かつ制御はユーザーに戻る。

サーチ結果の表示を要求するための手続が、図23に示されている。この手続は、ユーザーが表示ボタン179をクリックするときステップ316において開始される。それから、この手続は、図23に示されたステップ317に移動する。このステップにおいて、このシステムは、質問をし、かつその質問結果を得る。このステップは、図25に詳細に示されていない。質問結果が得られた後、この手続はステップ318に移動し、そして、一例が図24に表示されているサー

チ結果ウィンドー299を表示する。

再び、図23を参照すると、この手続の次のステップはステップ319である。(ユーザーが関連した表示階数ボタン298、299等をクリックした結果として)表示階数194に規定された各属性に対して、表示コラムが作られる。表示コラムは、階数ボタン194により規定された階数において左から右に作られる。それから、この手続はステップ320に移動し、そして、質問結果における各部品に対して、規定された属性に対する属性値が個々の表示コラム内に表示される。それから、ステップ321において、制御はユーザーに戻され、そしてリトリバー130は、別のコマンドを待つ。

図25を参照すると、質問をするための手続が詳細に例示されている。この手続は、ステップ322で開始する。このシステムは最初、質問がローカル質問か或いはグローバル質問であるかどうかをステップ323において決定する。もし質問がローカルのものであるならば、リトリバー130は、ステップ325に示されるように、選択されたクラス内に部品のリストを作る。もし質問がローカルのものでないならば、リトリバー130は、ステップ324に示されるように、選択されたサブツリー内に包含される部品の全てのリストを作る。いずれの場合に、このシステムは、ステップ326に進んで、処理されていない部品がリスト内に残るかどうかを決定する。もしその答えがyesならば、フローはステップ328に進み、そしてこのシステムは、リスト内の次の部品を得る。それから、この手続は、全ての属性セクタが相当する部品パラメータに一致するかどうかを判断するためにステップ329に進む。もしその答えがnoならば、この手続は、ステップ326に戻る。もしその答えがyesならば、この手続はステップ330に進む。ステップ330において質問結果に部品が付加され、そしてフローはステップ326に戻る。ステップ326を参照すると、もし処理されていない部品がリスト内に残っていないならば、そのときこの手続は、ステップ331に進み、そして質問結果及び部品カウントに復帰する。

図24を参照すると、サーチ結果ウィンドー299は、サーチにおいて発見された部品のための選択された属性を表示する。部品番号属性336は、この例においては階数ボタン298が最初に選択されたので、最左コラム内に現れる。仕

上げボタン299が第二に選択されたので、仕上げ属性337が第二に現れる。同様に、ヘッドスタイル属性338、ヘッドリセス属性339、長さ属性340、及び記述属性341が、階数ボタン194によって指示された階数で表示される。例示された例において、このサーチは、発見部品172を単一部品に限定した。もし、サーチ結果において1以上の部品が得られたならば、残りの部品はサーチ結果ディスプレイウィンドー299内の追加の行内に表示され、かつ、この表示は階数ボタン194により指示された階数で属性に従いソートされるであろう。

本発明の重要な性能最適化は、ネットワークリソースの使用を最適化するための質問結果の管理に関し、それによって、知識ベースサーバー132への効果的なアクセスを、ワイドエリアネットワーク2103を通して可能にし、かつこれは典型的には、ローカルエリアネットワーク100よりもかなり低い伝送速度及びデータを有している。これは、図222におけるフローチャートに示されるように達成される。ステップ2130においてスクロールされたリスト352をいずれかの方向にスクロールするユーザー要求にตอบสนองして、スクロールされている方向に部品情報を包含するバッファは、テスト2131で調べられて、スクロール要求が現在バッファ内にない部品情報に対する必要性を生じかどうかを判断する。もしそうならば、そのとき、知識ベースサーバー132から追加の情報を要求することなく、情報の1つの追加の表示ページのスクロールを可能にするために、質問結果からの十分な部品情報によって再び満たされる。ディスプレイをスクロールした後、部品情報は、ステップ2132において表示バッファから表示され、かつステップ2133において、制御はユーザーに戻される。この様に、もし全質問結果が最初にサーバーに伝送されたならば被ったであろうネットワーク伝送コストは避けられ、ワイドエリアネットワーク2103がローカルエリアネットワーク100の実際の代換えとなる点にまで応答時間をかなり改善する。この最適化はまた、全ネットワークトラフィックを減少させ、かつ典型的に質問システムで発見されるように質問結果において表示することのできる部品数を制限する必要性をなくする。

図24を参照すると、サーチ結果ウィンドー299は、部品情報ボタン342

を含んでいる。部品情報ボタン342の作動により開始された手続は、図26に示されている。ステップ332において、ユーザーは部品情報ボタン342をクリックする。このシステムはステップ333に進んで、図27の部品情報表示ウィンドー351に示されるように、部品のルートクラスから所有クラスにアウトラインフォーマット350でクラスバスの表示を発生する。図26を参照すると、手続フローはステップ334に進んで、属性名353及び値354を包含するスクロールリスト352の表示を発生する。

図27は、部品情報ウィンドー351を示している。これらの属性のための属性名353及び値354は、スクロールリスト352内に表示されている。図27に示された部品情報ウィンドー351は、OKコマンドボタン356を作動することにより閉じることができる。図26を参照すると、そのとき、手続はステップ335に進み、そして制御はユーザーに戻される。

サーチ結果ウィンドー299は、ユーザーアクションコマンドボタン343を含んでいる。このユーザーアクションボタン343は、他のユーザーアプリケーション又はソフトウェアプログラムを起動するために使用される。これは、このシステムから直接他のアプリケーションへの透明なアクセスを提供する。ユーザーアクションコマンドボタン343は、サーチ結果ウィンドー299における部品がその部品のための行番号をクリックすることにより選択されるときアクティブになり、そして、ユーザーアクションはその部品と関係している。

例えば、選択された部品のために描く実際の部品を見ることが望まれるかもしれない。CAD又はビューアーアプリケーションは、ボタン345をクリックし、そしてそれからプルダウンリスト344に含まれる所望のアプリケーションをクリックすることにより、アプリケーションのプルダウンリスト344から選択することができる。所望のユーザーアプリケーションが最初に選択され、それからユーザーアクションコマンドボタン343が作動して、アプリケーションを作動させ、かつ指定したファイルをオープンさせる。ユーザーアクションは、システムアドミニストレータにより定義することができる。

図28は、ユーザーアクションを起動するために使用される手続のフローチャートを示している。ステップ365において、ユーザーは、サーチ結果ウィンドー

ー299上のリスト344からユーザーアクションを選択する。それから、フローはステップ366に進み、そしてシステムは、関連したユーザーアクション定義をチェックすることによりユーザーアクションへの独立変数を探索する。

ステップ367において、このシステムは、ユーザーアクション定義において規定された部品属性336、337、及び340から満たされたパラメータによってコマンドラインをフォーマットする。ステップ368において、ローカルプロセスが、ユーザーアクションが完了しかつプロセスが終了するまで、フォーマットされたコマンドライン及びブロックを使って実行される。最後に、ステップ369において、制御はユーザーに戻される。

図24に示された例において、マイクロソフトウインドズライトプログラム344が選択された。ユーザーがユーザーアクションボタン343を作動するとき、ライトプログラム344がスタートする。この例において、部品番号336は、ライトプログラム344に通される。図29は、ライトプログラム344がスタートし、ここで、部品番号情報がライトプログラムに通されたときのユーザーアクションディスプレイスクリーン355を示している。

サーチ結果ウインドー299は、ソートコマンドボタン348を含んでいる。例示された例において、このボタン348は、1つのみの部品が表示されるので、薄暗くされる。複数の部品がサーチ結果ウインドー299内に表示されるとき、ソートコマンドボタン348が作動されて、表示情報を異なるように再ソートすることができる。

サーチ結果ウインドーは、プリントコマンドボタン347を含んでいる。このボタン347を作動すると、その部品からハードコピープリントを発生させる。これは、例えば、ユーザーがその部品について追加のサーチをすることを望むとき便利である。

サーチ結果ウインドー299は、アプライコマンドボタン346を含んでいる。このボタン346は、選択された部品のための属性値を、部品使用ウインドー170内のサーチ基準フィールド177にコピーするために使用される。これらの値がコピーされた後、サーチ結果ウインドー299は閉じる。アプライコマンドボタン346を作動させた結果が図31に示されている。これは例えば、パラ

メータの1つをゆるめる別の質問を行うために便利に使用することができる。

アプライコマンドボタン346が作動するとき実行される手続が、図30に示されている。この手続は、ユーザーがアプライコマンドボタン346を作動するときステップ370で開始する。それから、ステップ371が実行され、そして新たな質問が、その質問クラスとして選択された部品所有クラスによって作られる。ステップ372において、適切な属性セレクトが、部品のためにそれぞれ定義された属性のための質問に付加される。

部品仕様ウィンドー170が、現質問のクラス240にオープンなクラスアウトライン248によってステップ373において表示される。それから、ステップ374において、属性セクタ242、281等が現質問のために表示される。それから、このシステムは、部品カウント172及びディスプレイ170を更新する。ステップ375において、制御はユーザーに戻される。

或いは、サーチ結果ウィンドー299は、クローズボタン349を作動することにより閉じることができる。

必要なアクセス権を有するユーザーは、編集コマンドボタン180を作動することにより知識ベース内の部品情報を編集することができる。実行されるこの手続は、図32に示されている。ステップ376は、編集ボタン180が選択されるとき実行される。ステップ377において、質問が現サーチ基準177に基づいて実行され、かつトリガー130は、質問結果を得る。それから、質問結果はステップ378においてスプレッドシートフォーマットで表示される。

ステップ379において、システムは部品移動、削除、及び属性編集要求を取り扱う。

ユーザーがサーチ結果ウィンドー299においてソートボタン348を作動するとき実行されるソート手続は、図33のフローチャートに示されている。ステップ380は、ユーザーがソートボタン348をクリックするとき実行される。それから、このシステムは、ステップ381で述べたように、ソートダイアログボックス386を表示する。

ソートダイアログボックス386の例が図34に示されている。属性387〜392は、図33のステップ381に従って属性コラム393内に表示される。

ダイアログボックス 386 はまた、ソートキーコラム 394 及びソート階数コラム 395 を含んでいる。ソート階数コラムは、各属性 387-392 のための上昇階数又は下降階数をユーザーに選択させる適切なボタン 396 (その 1 つのみが示されている) により作動されたプルダウンメニューを包含している。これは、図 33 に示されたステップ 381 に記載されている。

ユーザーは、並べ替え手続によって、部品リストを英数字あるいは数字の順に再編成することができる。ユーザーは一つあるいはそれ以上の属性値 387~392 を基にして部品リストを昇順あるいは降順に並べ替えられる。並べ替えキーはユーザーが最初に並べ替えようとする属性 387~392、次に並べ替えようとする属性、第 3 番目に並べ替えようとする属性などを特定する。

例えば、ユーザーが部品リストを持っていて、一つの属性 392 によって並べ替えた部品リストを得ようとした場合には、ユーザーは最初の並べ替えキーとしてその属性 392 を選ぶ。最初の属性 392 が重複した値を持っている場合には、ユーザーは並べ替えの第二の属性 389 を選ぶことができる。表 3 に示す例では、長さの属性 392 が第 1 のキーで順は昇順である。主な材質属性 389 は並べ替えキーとして選択されていない。

表 3

長さ	主な材質
. 5 (インチ)	リン
. 5 (インチ)	亜鉛
. 5 (インチ)	リン
. 75 (インチ)	リン
. 75 (インチ)	亜鉛
. 75 (インチ)	リン

表 3 の例に注意すると、長さの欄に重複した値があり、そのために材質の欄の値はランダムである。ユーザーがこれらの値を最初のキーとともに並べ替えようとするときには、第 2 の並べ替えキーとして主な材質属性 389 を選択しなければならない。この例でユーザーが両属性 392 と 389 について既定の並べ替え

順、すなわち昇順とする。表4に示すものがこの種の並べ替えをした結果である。

表 4

長さ	主な材質
. 5 (インチ)	リン
. 5 (インチ)	リン
. 5 (インチ)	亜鉛
. 7 5 (インチ)	リン
. 7 5 (インチ)	リン
. 7 5 (インチ)	亜鉛

並べ替え順として昇順を選ぶことで、列挙された属性を並べ替える場合、属性値の順を属性のタイプで決まる方法で並べ替えられるようにする。「定義のない」値を最初にリストし、それから残りの値についてそれらが枠組みに表われているのと同じ順序でリストにする。文書属性を並べ替えるときには、「定義のない」値の文書属性を最初にリストにして、それから数字の値のもの、それから文書のもものをASCII順で並べ替える。数字属性を並べ替えるときには、「定義のない」値を持った数字属性をまずリストにし、それから測定単位を基にして数字の値をリストする。プールの属性を並べ替えるときには、「定義のない」値のプールの属性を最初にリストにし、それから真の値の属性、その次に偽りの値の属性とする。

並べ替え順として降順としたときは、属性値の順は逆となる。

ユーザーが使おうとする並べ替え順を確立するために、ユーザーは並べ替えコマンドボタン384を選択し、次に並べ替えダイアログボックス386からユーザーが最初に並べ替えようとしている属性392を選択して、セットコマンドボタン398を選択する。セットコマンドボタン398を選択することで属性392のキー394と並べ替え順395をセットする。属性392のキーフィールド394をダブルクリックすることで並べ替えキー394と並べ替え順395の両方をセットする。

キー394と並べ替え順395を取り消すには、取り消したい属性392を選んでセットコマンドボタン398を選択する。選んだ属性392のキー394と並べ替え順395が取り消される。図33のステップ382にあるように、ユーザーは取消しコマンドボタン397を活性化することで人力を取り消すことができる。ユーザーがそのようにしたら、フローはステップ385に飛んでユーザーがコントロールできるようになる。

ユーザーが並べ替えようとする属性387～392のすべてを選んだ後、OKコマンドボタン399を活性化することができる。このことで図33のステップ383にフローは進む。ステップ383で、クエリーの結果は必要な複合並べ替えキーに従って並べ替えられる。ステップ384によって、並べ替えダイアログボックス386が閉じて、ユーザーが選んだ並べ替えた部品情報を持った検索結果ウィンドウ299が再表示される。そしてステップ385で、ユーザーがコントロールできるようになる。

ユーザーが更なる情報を持って部品を更に分類する、あるいは重複した部品があったり、あるいは部品を一つの分類から他の分類に移動させる必要があるなど、ユーザーの知識ベースにある部品を編集する必要が出てくることがある。編集コマンドボタンはそのユーザーがその画面にアクセスする権利のある場合に部品仕様ウィンドウに現れる。システム管理者はこの画面へのアクセス権を設定することができる。ユーザーが編集を必要とする部品を探すことを助ける2つの画面がある。その一つは「定義のない」属性を持った部品を探すものである。たのものは完全には分類されていない部品を見つけるために部分的なクエリーコマンドを使うものである。

部品編集ウィンドウ1019で、ユーザーは属性値の編集をしたり、部品を一つの分類から他へ移動させたり、部品を削除したりすることができる。部品を編集するのに、ユーザーは部品を指定するのに使ったのと同じ手順を使う。特定の属性値1056を持った部品を指定するのに加えて、ユーザーは特定の属性1060について値（定義なし）を持たない部品を探すことができる。定義なしの属性値を持った部品の場所を探すのに、特定の属性166について定義なしチェックボックス165を選ぶ。図6を見よ。定義なしチェックボックス165にチ

エックマークが付いている場合には、それが選ばれたのであり、定義された属性 166 を持たない部品をユーザーが探していることを示している。ユーザーが知識ベースのデータを編集していて、現在定義なし 1060 の属性 166 の位置を探そうとする場合は、「定義なし」を使って、それらの部品を探し適当な値が含まれるように知識ベースを更新することができる。

ユーザーは特定の属性値とともに「定義なし」を選ぶ場合には、検索条件の一部に or 条件を入れる。この例では、特定の値を持った部品とその属性には値のない部品の位置を探し出す。定義なしのチェックボックス 165 は検索条件フィールド 177 の右側にある。ユーザーは水平スクロールバーを使ってチェックボックス 165 を見える場所へ動かすことができる。

ユーザーが分類 174 と副分類 196、197、198 および 199 を選択して部品を特定し、属性検索条件 177 を入力し、表示順序 194 をセットしたら、編集コマンドボタン 180 を選ぶことができる。

図 35 に、部品を編集するときに従う手順を示すフローチャートを示している。例えば図 21 を参照して、部品編集にアクセス権のあるユーザーは編集ボタン 180 を活性化して、図 36 に示した部品編集ウィンドウ 1019 を出すことができる。図 35 の第 1 ステップ 1012 は、部品編集ウィンドウ 1019 から編集する属性と部品を選ぶことである。ユーザーが属性 1051 の新しいあるいは更新された値 1061 を入力すると、システムはステップ 1013 でパラメータの入力を受け入れる。属性が列挙した属性 1051 の場合には、図 37 に示したようにプルダウンリスト 1062 が示されて選択できるようになる。図 35 のステップ 1014 で、他の部品を編集するかどうかを決める。編集するものがない場合には、フローはステップ 1017 に進む。システムは、部品表示 1020 と部品編集ウィンドウ 1019 を更新して編集された値 1061 にする。システムはステップ 1018 に進み、ユーザーがコントロールできるようにする。

ステップ 1014 で、編集するものがまだある場合にはフローはステップ 1015 に進み、システムは次に選択された部品を取り入れる。ステップ 1016 で、システムは次に選択された部品パラメータをユーザーの入力値 1061 にセットする。そしてステップ 1014 に戻る。

図 38 は部品を削除する手順を示す。ステップ 1021 で、ユーザーは部品編集ウィンドウ 1019 から削除すべき部品を選択する。それから、部品削除コマンドボタン 1026 をクリックする。ステップ 1022 で、削除すべきほかの部品が残っているかどうかを決める。答えが yes の場合には、フローはステップ 1023 に進み、システムは次に選択された部品に進み、それをクエリー結果と知識ベースから削除する。そしてフローはステップ 1022 に戻る。それ以上の部品を削除しない場合には、フローはステップ 1024 に進み、システムは部品編集ウィンドウ 1019 に更新したクエリー結果を表示する。そしてフローはステップ 1025 に進み、ユーザーがコントロールできるようになる。

図 39 は部品を移動させる手順のフローチャートを示す。その手順は、ステップ 1032 に示したように部品編集ウィンドウ 1019 から移動させる部品を選択することから始める。あるいは、ステップ 1033 にあるように、部品編集ウィンドウ 1019 上の分類階層に従って行き先の分類を選択することから手順を始める。例えば図 40 に描かれているように、部品移動コマンドボタン 1027 を活性化する。

図 39 を参照して、手順はステップ 1034 に進み、移動させるべきほかの部品があるかどうかを決める。移動させるべき他の部品がない場合には、フローはステップ 1042 に移り、システムは部品編集ウィンドウ 1019 にクエリー結果を表示する。そしてフローはステップ 1043 に進み、コントロールはユーザーに戻る。

図 39 のステップ 1034 に戻って、他の部品を移動させることにした場合には、フローはステップ 1035 に進み、システムは次に選ばれた部品に進む。ステップ 1036 で、ユーザーが無条件移動をするかどうかを決める。答えが yes の場合には、フローはステップ 1040 に飛ぶ。そしてシステムはユーザーが選んだ行き先の分類に部品分類をセットする。なにかのパラメータがない場合や属性が定義なしとなっている場合には、フローはステップ 1041 に進み、クエリーの結果から移動した部品を削除する。フローはステップ 1042 に進み、部品編集ウィンドウ 1019 にクエリー結果を表示する。

ステップ 1036 で、ユーザーが無条件移動をしないことにした場合には、フ

ローはステップ1037に進み部品パラメータの属性が行き先の分類から無くなっているかどうかを決定する。答えがnoの場合には、フローはステップ1040に進み、上に述べたように続けられる。

行き先の分類から無くなっている部品パラメータの属性があるとステップ1037でした場合には、フローはステップ1038に移る。システムは、移動してなくなってしまうパラメータのリストを得て、デスプレイ116にそれを表示してユーザーにそれを示す。もしもユーザーがパラメータが削除されるという警告を無視するか部品を無条件で移動させようとした場合にはフローはステップ1040に移り上で述べたように処理される。もしもユーザーがパラメーターが削除されるという警告を無視したくないかあるいは無条件で部品を移動させたくない場合にはフローはステップ1034に戻る。

部品編集プロセスは部品編集ウィンドウ1019 (図40に示したように) の説明を参照して更によく理解することができる。ユーザーが分類174と吹く分類196、197、198および199を選択して部品を特定し、属性検索条件177を入力し、表示順序194をセットすることで、編集コマンドボタン180を選択して部品を編集することができる。このコマンド180を選ぶことで部品編集ウィンドウ1019を現すことができる。部品編集ウィンドウ1019の上の領域1052は分類ツリー1044を有し、ユーザーが編集している部品の道筋と分類定義を浮かび上がらせてしめす。ウィンドウ1019の底の領域1053は、ユーザーが編集することを選択した部品1020を示している。その部品はスプレッドシートアプリケーションで使ったものと同様な表で示される。部品属性1049、1050、1051等および属性値1055、1056、1057等は、ユーザーが前に部品仕様ウィンドウ170で決めた表示順序で左から右に現れる。値を使うのに、エンターボックス1063をクリックする。新しい値を削除するには削除ボックス1064をクリックする。

部品編集ウィンドウ1019の一番上の領域1052には分類定義1044を含み、それは編集を選択した部品の道筋と分類定義を示す分層ツリーを持っている。ウィンドウ1019には水平分割バー1047を持っておりウィンドウを2つの領域に分割する。ユーザーは分割バー1047を上にあるいは下に動かして

一つの領域1052あるいはもう一つのもの1053を大きくできる。部品編集ウィンドウ1019には編集領域1046と呼ばれる領域がある。属性値1051を選択した後は、(図36参照)編集領域1046に文書ボックスすなわちリストボックス1054が現れるので変更を加えることができる。部品は表1020の行1048として現されて、表1020の各々の行1048は番号が付けられている。ユーザーは行番号を使って情報を得たいものや移動あるいは削除をしたい部品を選択することができる。属性1049、1050、1051等は列のヘッディングで、属性値は行である。

図40を参照して、並べ替えコマンドボタン1029を活性化して、ユーザーが入力する並べ替え順に従って部品を再編成することができる。部品情報コマンドボタン1028を活性化して、選択された部品のすべての部品情報(分類定義とすべての属性)を表示することができる。プリントコマンドボタン1030を活性化して、部品のリストを印刷することができる。ユーザーが部品を選択した後で削除コマンドボタン1026は活性化して、知識ベースから不要になった部品を削除するのに使うことができる。クローズコマンドボタン1031を活性化して、部品編集ウィンドウ1019を閉じて部品仕様ウィンドウ1070に戻ることができる。

部品を新規に作成するのに、部品を特定するのに使ったのと同じ手順に従う。指定した部品が見つからなくてまた受け入れられるような代替品がない場合には知識ベースに新しい部品を追加することができる。

知識ベースに新しい部品を入れることを決めた場合にはユーザーはその部品を完全に特定しなければならない。望ましい実施態様としては、リーフ分類201までのすべての分類を選択してすべての必要な属性203についての値を入力して完全な部品仕様が決められる。もしもユーザーがリーフ分類201を選択しなかったり完全な属性203を入力しなかったら部品を追加できない。部品を新規作成するには、望ましい手順は、ユーザーができるだけ多くの属性値203を入力して、できるだけ完全な仕様を部品に与えることである。

ある属性は部品を追加する前に必要となる。必要とされる属性は属性アイコンのすぐ左に必要なアイコン263を持っている。新規作成コマンド181を選択

する前に、各々の必要な属性に属性値を入力する必要がある。プロテクトのかかった属性は属性アイコンのすぐ左にプロテクトされたアイコン 191 を持っている。ユーザーがリーフ分類 201 を選択してすべての必要な属性を入力したら、新規作成コマンドボタン 181 を選択することができる。新規作成コマンドボタン 181 を選択することで部品を知識ベースに追加して見つかった部品 172 を更新し部品カウントを 1 とする。

上の説明はウィンドウズクライアント 112 を参照しているが、システムはそれに限定されない。

B. 知識ベースクライアント

知識ベースクライアント 131 は API 143 を通してクライアントアプリケーション 130、133、144 に知識ベースサービスをする一連の C++ ライブラリーである。そのサービスは極地的なものとして知識ベースサーバー 132 に遠くからアクセスするものがある。ウィンドウズで動くクライアントアプリケーションとして、知識ベースクライアントは一つまたはそれ以上のウィンドウダイナミックライブラリ (DLL) からなり、それはウィンソック DLL を使い知識ベースサーバー 132 とレジストリサーバー 141 にネットワークアクセスをする。

C. 知識ベースサーバー

知識ベースサーバー 132 は知識サーバー 103 にアクセスしたり、検索したり、それを更新したりする UNIX サーバプロセスである。知識ベースサーバー 132 は一つまたはそれ以上の知識ベース 103、105 を使うことができる。

。

1. ダイナミッククラスマネジャー

ダイナミッククラスマネジャー 134 は知識ベースサーバー 132 のソフトウェアサブシステムであり、枠組みやデータを扱う。ダイナミッククラスマネジャー 134 は分類、属性、単位、例示情報を蓄える能力があり、それらはダイナミックに修正することができる。ダイナミッククラスマネジャー 134 は C++ ライブラリとクラスからなり、分類、属性、事例、パラメータ、単位ファミリー、単位、実行時の仮属性を伝承、アクセス、創造、削除、修正する働きがある。

ダイナミッククラスマネージャー 1.3.4 の性能はユーザープログラマーが API 1.4.3 の働きを通して変更することができる。

ダイナミッククラスマネージャー 1.3.4 の知識ベース、今後時々「知識ベース」と呼ぶことがある、は分類、属性、単位、パラメータ値のある事例、これらのオブジェクトの間の関係の集合体である。ダイナミッククラスマネージャー 1.3.4 で、分類は特定のタイプのオブジェクトを決める。分類は属性を決める。属性はタイプがあり、オブジェクトの性質を決めるのに使われる。分類は他の分類から派生することがある。この場合、その分類はその祖先の属性を受け継ぐ。知識ベースは分類の事例を持つ。事例で決められる属性値はパラメータである。分類、属性、事例およびパラメータの考えを説明するのに例として犬を使う。「犬」の語は、分類のようなものである。犬で一連の性質や属性の持ったよく似たものの一つのグループを示す。犬の属性は色や血統や名前のようなものである。分類や属性ではいかなる特定の犬をも特定しないが、それを説明するのに便利である。犬の事例は属性の値であるパラメータを持っている。例えば、犬の色はベイジュであり、血統はゴールデンレトリバーであり、名前はサンディである。

分類の間には関連がある。「犬」の分類はより大きな分類「哺乳動物」の一部である。「哺乳動物」の分類は「犬」よりも特定するものではない。それはオブジェクトである「犬」よりも伝える情報が少ないが、「哺乳動物」の情報総てが「犬」に適用できる。「犬」は「哺乳動物」の部分集合であり、この関係は副分類である。「犬」は「哺乳動物」の分類の副分類である。この副分類「犬」は更に小さな分類、大きな「犬」、小さな「犬」などに分けることができる。副分類の概念は二つの分類の間の親子の関係を暗示している。「哺乳動物」は親であり、「犬」は副分類である。「犬は哺乳動物から派生した」との言い方はその関係を説明するのに使われる。

副分類「犬」はその親分類の属性を受け継いでいる。すべての「哺乳動物」は色を持つので、属性色は「哺乳動物」分類の一部である。「犬」分類はその親から属性色を受け継いでいる。

ルート分類は特別なものであり、その親はない。それはすべての分類が派生しはじめる分類である。ここの説明で、分類階層を説明するグラフが描かれていた

とすると、ルート分類はその図のトップに位置する。副分類はルート分類から枝分かれしており広がった道筋になっており、木を逆さにしたように見えるグラフになっている。分類のすべてのグループでツリーになっており、その一番上にあって親を持たない特別な分類がルートである。

ダイナミッククラスマネジャー 1 3 4 でサポートされていて使用できるタイプの属性の一つは数字である。数字属性は現実には測定できる量をいうのに使われる。そのような測定は単に数値だけから構成されているのではなく、ある関連した単位を持っている。ダイナミッククラスマネジャー 1 3 4 はユニットマネジャー 1 3 8 と一緒になって数字属性と一緒に使われる種々のタイプの単位に関する情報を持っている。ダイナミッククラスマネジャー 1 3 4 はユニットマネジャー 1 3 8 を使って単位間の変換を行うことができる。システムが理解できる単位は種々の単位ファミリーにグループ化されている。これらの単位ファミリーと単位は実行時に変換される。ダイナミッククラスマネジャー 1 3 4 はまたダイナミックユニットマネジャー 1 3 8 を持っている。

「枠組み」の語は分類、属性、単位、単位ファミリーの並んだものを指す。事例のない知識ベースは枠組みである。これはダイナミッククラスマネジャー 1 3 4 で使われる種々のオブジェクトについての以下のより詳細な説明によって更によく理解できるであろう。

分類は本発明において枠組みの中の最も基本的なオブジェクトである。分類は関係するオブジェクトの集合体である。この例では、分類は 8 個ないし 9 個の成分を有することがある。分類は枠組みのオブジェクトである。上に述べたように、枠組みは分類や属性や単位や単位ファミリーやそれらの関係の集合体である。すべての分類はルート分類 1 7 3 を除いてそれが派生しているただ一つの親を持っている。ルート分類 1 7 3 は親を持たないただ一つの分類である。ルート分類 1 7 3 は決して消すことのできない他の特別な性質を持っている。ある分類がその親から派生している結果としてその分類はその親が持つすべての性質を持っている。これらの性質は属性と呼ぶ。属性はその親分類から受け継いだものである。

ある分類は零かそれ以上の副分類を持つことがある。分類はその副分類の親で

ある。副分類は親のある分類である。そこで、ルート分類173は副分類ではない。ある親分類の副分類はある決まった順を持っている。その順序は永続的なものでダイナミッククラスマネジャー134は知識ベースを閉じているときも再開したときもその順序を保つ。

分類はその副分類、その副分類の副分類などからなる一連の後継者を持つ。副分類が零か後継者が空の分類をリーフ分類201と呼ぶ。サブツリーはある分類とそのすべての後継者からなるセットである。そのサブツリーはその分類にルートを持つという。ある副分類はまた、その親、その親の親、等々ルーと分類173を含めたセットである一連の祖先を持っている。同じ親からなる分類は時々きょうだいと呼ぶことがある。

副分類からその親にたどっていくことは時々ツリーを上と呼ぶことがある。親からその副分類に移動することをツリーを下と呼ぶことがある。それゆえ、枠組みのルート分類173はツリーのトップにあり、ツリーの最も底にあるオブジェクトは典型的なリーフ分類201である。

図41は分類800の内部オブジェクトを示している。この枠組みでは、分類は親のハンドル801である。親を持たないルート分類173の特別な場合を除いて、すべての分類オブジェクト800はその親のハンドルを示す情報を持っている。その場合にはこの場所に零が入れられる。ハンドルはオブジェクトを参照するものである。親のハンドル情報801はハンドルマネジャー137で使われてその分類800の親分類である蓄えられている分類オブジェクトを特定する。

分類オブジェクト800は副分類リスト802を持つ。副分類リスト802はハンドルの列を持ち、それはハンドルマネジャー137で使われてその分類の副分類オブジェクトを特定する。本発明の内部表示では、リストは無限なく大きくすることができダイナミックなものである。必要な格納スペースは制限されない。

分類オブジェクト800は性能に大きな欠点を生じないで大きなデータベースでは大量の副分類を持つことができるので、データベース構造に柔軟性と能力を与える。

分類オブジェクト800は属性リスト803を持つ。ハンドルマネジャー13

7は属性リスト103に格納されている情報を使って分類オブジェクト800が持っている属性を特定する。

分類オブジェクト800はまた地域的な事例リスト804を持っておりそれはハンドルのリストである。図41のフィールド805は分類名、すなわち分類を特定する文章、を格納している場所を指し示すものである。

フィールド806は分類800のハンドルを格納するのに使われる。フィールド807は分類コード、すなわちそれが一次か二次あるいは集合的なものかなど、を示すものを格納している。

分類オブジェクト800はサブツリー事例カウント808を持っている。サブツリー事例カウント808は、項目すなわち分類800のすべての後継者にある事例の合計数、すなわち分類800の事例の全数、分類800の副分類のすべて、副分類の副分類総などを数字で示すものである。例えば図10を参照すると、事例カウント808は、見つけた部品172フィールドを作り出すのに使われて、それは部品マネジメントウィンドウ170に表示される。このようにして、知識ベースのツリー構造を使って、副分類を選んで聞くと、現在の分類のサブツリー事例カウント808を検索しその情報をレトリバー130へ伝えることによって、見つけた部品172がツリーのどこにあってもその数が直にわかるようになっていく。知識ベースが更新されるとサブツリー事例カウント808は更新されるので、データベースのツリー構造を使っているときに見つけた部品172をその場で処理する必要が無い。

再び図41を参照して、分類オブジェクト800は望ましくはメタパラメータリスト809を有している。リンクしている情報を指し示すのに用いる。それは例えば、分類800で示される部品のタイプを図で表している。ファイルの名前や、データを伝えるのに使われるシソーラス情報や他の伝承情報などである。図42は共通のリスト810の例を示す。データの変化量がオブジェクトと結びついているときには、クラスマネジャー134はハンドルのリスト、浮動小数点値のリスト、文字列のポインターのリストなどを使う。リストの例は項802、803、804、809であり、リスト810は単純な整数である。リストオブジェクト810はリストデータ811の最初815を指しているポインター8

12を有する。またリストオブジェクト810はリストデータ811に現在許されている大きさを示すフィールド813を持つ。またリストオブジェクト810は現在使われているリストデータ811の量を示す情報を含んだフィールド814を持つ。

リストデータ811は実際の値のリストを含む。この例でリストの最初の項815は値「5」を持つ。同様に、項816、817、819、820、821は他の値を有する。リストの項822、823、824、825、826は現在使われていないので零にセットされている。この例では、リスト813に現在許されている大きさは12である。リスト814で使用している量は7で、これはリストの最初の7項が有効であることを意味している。

図43は属性データのデータ構造を示している。

属性オブジェクト827は実施態様にあるように少なくとも6フィールドを持つ。最初のフィールド828は、属性の名前であるASCII文字列を有する外部名に対するポインタを持つ。また、属性オブジェクト827はこの属性オブジェクト827のハンドルを持ったフィールド829を有している。また、属性オブジェクト827は、この属性827を定義している分類のハンドルを持ったフィールド830を有している。第4のフィールド831は、この属性がその分類を定義するのに必要なものかどうかを示すブーlean表示である。第5のフィールド832はこの属性がプロテクトされているかどうかを示すブーleanのフィールドである。例えば、図6で「部品数」属性176がプロテクトされている。これはプロテクトアイコン191で示されている。図43の属性オブジェクト827のデータ構造では、この情報はフィールド832に格納されている。属性オブジェクト827はまたメタパラメータリストのフィールド833を持つ。列挙された属性にはフィールド828～833、これらはまとめて属性データ834と示されている、に加えてエニユマレータハンドルのリストであるフィールド835がある。

ブーleanの属性値の場合には、フィールド828～833だけが使われて、それは図43に属性データ834としてまとめて示されている。

数値属性はフィールド828～833、これらはまとめて属性データ834と

示されている、に加えてこの数字属性の単位ファミリーのハンドルを含むフィールド833がある。

ストリング属性の場合とストリング列属性の場合にはフィールド828～833を含む属性データ834だけが含まれている。

これらのデータ構造をダイナミックマネジャー134で使う例は、(図7参照)その分類の閉じられたフォルダーのアイコン189をクリックしてその分類を選択することである。分類が開かれると、ダイナミッククラスマネジャー134は分類オブジェクト800をチェックして属性リスト803を検索する。属性リスト803に格納されているハンドルはハンドルマネジャー137に伝えられる。ハンドルマネジャー137はその分類の各属性827の実際のメモリアドレスに戻る。ダイナミッククラスマネジャー134は属性オブジェクト827の外部名のポインター828を使って、その属性の外部名のキャラクターストリングテキストを検索する。そうしてそのASCIIテキスト情報はAPI143を通して伝えられるので、レトリバー130に与えられて、ディスプレイ116に示される。

図44はエニユマレーターオブジェクト841のデータ構造を示している。エニユマレーターオブジェクト841は3個のフィールドを持つことができる。第1のフィールド842はエニユマレーターオブジェクト841の外部名のポインターである。第2のフィールド843はエニユマレーターオブジェクト841のハンドルである。第3のフィールド844はメタパラメータリストである。ハンドルは他のオブジェクトをエニユマレーターオブジェクト841に結びつけるのに使われる。この構造の利点はオブジェクトの外部名に変更することが望ましいときには容易に知識ベースを修正できることである。外部名を示すのに使われるASCII文字列に変更するだけでそのような変更をすることができる。他のすべてのオブジェクトは単にハンドルマネジャー137で使われるハンドルで、ダイナミックマネジャー134に実際の外部名を与える。

図45は単位ファミリーオブジェクト845のデータ構造を示している。図45に示した例では、単位ファミリーオブジェクト845は4個のフィールドを持っている。第1のフィールドは単位ファミリーオブジェクトの外部名846のボ

インターである。第2のフィールド847は単位ファミリーオブジェクト845のハンドルである。第3のフィールド848は単位ファミリー845に含まれる単位ファミリーハンドルのリストである。フィールド849はローカルな単位のハンドルリストである。

単位は数字パラメータの測定系である。単位ファミリーは数字属性に使われる単位の集合体である。単位ファミリーハンドルは単位ファミリーを参照するものである。単位ファミリー名はASCIIテキストで単位ファミリーを特定する。単位ハンドルは単位を参照するものである。単位名はASCIIテキストで単位を特定する。ローカル単位はこの単位ファミリー845で定義された単位である。

図46は単位のデータ構造を示している。単位オブジェクト850は5個のデータフィールド851～855を持つことができる。第1のフィールド851は単位の外部名のポインターである。単位オブジェクト850のハンドルは第2のフィールド852に格納されている。第3のフィールド853は単位ファミリーを定義するハンドルである。第4のフィールド854はメタパラメータリストである。最後のフィールド855は単位のタイプ（例えば実数、整数、または列挙した表）を示す。これら5個のフィールド851～855はベース単位データ856を含む。単位オブジェクト850がベース単位の場合には、他のデータは必要ない。これは図46の項862に示されている。単位オブジェクト850が列挙された派生単位867の場合には、それはベース単位データ856を含み、フィールド851～855を有する。列挙された派生単位867はベース単位のハンドル与えるフィールド858を持つ。他のフィールド856はその列挙リストが何行あるかについての情報を与える。フィールド860はASCII文字列からなるエニューレータのリストである。フィールド861はフィールド860のエニューレータのリストに相当する値のリストである。

単位オブジェクト850が実際の派生単位866の場合にはフィールド851～855を含むベース単位データ856を有する。更にそれはベース単位のハンドルを格納するフィールド863を有する。第2の別なフィールド864は実際の派生単位866を導くのに使う倍数ファクターである。第3の別なフィールド

865はオフセットであり、実際の派生単位866を導くのにベース単位に加算、減算をするものである。例えばベース単位850が摂氏温度であるとき実際の派生単位866が華氏温度であるとする、倍数ファクター864は9/5でオフセット865は32度である。

図47は単位ファミリーのデータ構造を示している。ダイナミッククラスマネジャー134はただ一つの包括的な単位ファミリーのハンドルリスト836を有している。そのリストの一つの要素837は単位ファミリー845のハンドルである。簡単にするためにハンドル837から単位ファミリー845まで矢印を引いてある。実際は、リスト836からのハンドル837はハンドルマネジャー137に伝えられ、ハンドルマネジャー137は単位ファミリー845の実メモリのアドレスを出す。それゆえ、ハンドルマネジャー137はハンドルをそのハンドルに関連したオブジェクトにリンクさせると言うことを理解すべきである。オブジェクトを参照するのにハンドルが使われたときにハンドルマネジャー137とのリンクが起こるということを理解しているので、説明を簡単にするために、ここではハンドルマネジャー137のことは除外する。加えて、この説明には不必要なデータフィールドやデータメンバーも図47から除いている。図47の例では、単位ファミリーハンドル848のリストは空である。ローカル単位ハンドル839の実際のリストは単位ファミリーオブジェクト845のリストオブジェクト849で指される。ローカル単位ハンドル839のリストに行って、ダイナミッククラスマネジャー134は必要な単位オブジェクトを探し当てることができる。例えば、リスト839の項857は実際の派生単位866を参照するハンドルである。この例では単位ファミリー845は「抵抗」であり、派生単位866は「キロオーム」である。派生単位866はフィールド863に格納されているベース単位のハンドルを有している。フィールド863に格納されているハンドルはベース単位を探し当てるのに使われる。この例では、ベース単位の名852は「オーム」である。派生単位オブジェクト866は倍数ファクター864を持ち、この例ではそれは1000である。このようにして、単位マネジャー138は倍数ファクター864を使って、派生単位「キロオーム」850に1000を掛けることでベース単位「オーム」866に変換することができる。

実派生単位オブジェクト 866 は単位ファミリー 845 を定義するハンドルを持つ。また、単位オブジェクト 850 は単位ファミリー 845 のハンドルを持ったフィールド 853 を有している。

図 48 は列挙された派生単位のデータ構造を示す。包括的な単位ファミリーハンドルリスト 836 はリスト中に項 837 を有し、それは図 47 を参照して説明したように、単位ファミリー 845 のハンドルである。しかし、この例では包括的な単位ファミリーハンドルリスト 836 はまた第 2 の単位ファミリー 845' のハンドルになる項をリスト 862 中を含む。第 2 の単位ファミリー 845' は名前 847' を持つ。この例に含まれている単位ハンドル 848' のリストは単位ファミリー 845 のハンドルを有する。単位ファミリーオブジェクト 845' はローカルな単位ハンドル 839' のリストを指すデータフィールド 849 を持つ。リスト 839' は列挙した派生単位オブジェクト 867 のハンドルである項 868 をリスト中を含む。この例では、列挙した派生単位オブジェクト 867 の名は「オームの表」852 である。フィールド 859 はこの列挙した派生単位オブジェクト 867 に含まれている行の数に関する情報を有する。フィールド 860 はエニユマレーター 869 のリストを指しており、それは列挙したリスト、この例では「10k」「11k」「12k」等である、からユーザーに選ばれた値をリストしている。リスト 869 は ASCII 文字列を含む。列挙された派生単位オブジェクト 867 では、フィールド 861 は実際の数値データ値 870 のリストを指している。リスト 869 の項目とリスト 870 の数値の間には 1 対 1 対応がある。図示した例では、リスト 870 は実際の数値 10000、11000、12000 などを含む。これらの値はこの例ではオームを示しているの、「10k」「11k」「12k」などのオームに対応する。もちろん列挙された派生単位オブジェクト 867 にはベース単位オブジェクト 850 のハンドルを有するフィールド 858 があり、それはこの場合には「オーム」の名前 852 を持つ。

図 48 はダイナミッククラスマネジャー 134 で使われて、図 16 に示したような標準値ウィンドウ 273 の表を表示するのに必要な情報をレトリバー 130 に与えるデータ構造を示している。図 16 で複数の標準値 275 は、図 48 に

示したリスト869に含まれているASCII文字の表示である。値275の一つがユーザーに選ばれたら、ユニットマネジャー138は図48で示したリスト870から相当する数値を選んでダイナミッククラスマネジャー134に与える。

図49は事例871と関連するパラメータ872のデータ構造を示す。事例オブジェクト871は4個のフィールド873～876を有する。第1のフィールド873はこの事例のオーナーの分類のハンドルである。第2のフィールド874はそれが持っている分類の事例リスト804の中でその事例ハンドルが最初に位置していた場所についてのハンドルである。第3のフィールド875はパラメータのリストであり、877に含まれている値を指す。第4のフィールド876は事例オブジェクト871のハンドルである。パラメータ877のリストは、その事例オブジェクト871が関連している種々の属性のパラメータに対する複数のポインターを含む。図49に描いた例では、リスト877は3個のエントリー878、879、880を持つ。リスト877の他の要素は明確にするために削除してある。リスト877のポインター878は関連するパラメータに関する情報を指す。パラメータ872のデータ構造は図50に詳しく描いた。

図50は、5個の違ったタイプ、列挙、ブーレ、数字、ストリング、ストリング列、のパラメータのデータ構造を示す。各パラメータオブジェクト872は属性ハンドル881を持つ。列挙オブジェクト888は属性ハンドル881とエニユマレーターハンドル882を有する。ブーレオブジェクト889は属性ハンドル881とブーレ値883を持つ。数字パラメータオブジェクト890は、属性ハンドル881と単位ハンドル884と値885を持つ。例えば、数字パラメータが10オームだったら、単位ハンドル884はオーム単位のハンドルであり、値885は10である。ストリングパラメータ891は属性ハンドル881のフィールドとASCII文字列のポインター886を持つ。ストリング列パラメータ892は属性ハンドル881とストリング列のポインターリストを指すフィールド887を有する。

図51は事例を持った枠組みの例である。この例で「エレクトロニクス」という名の分類で、副分類800'は「キャパシター」という名である。キャパシタ

一の副分類800'は「ケースタイプ」と呼ぶ属性827を持つ。この例では可能なケースタイプは二つあり、それらは「ケースA」と「ケースB」である。副分類キャパシター800'は「電解」という名の副分類800'を有する。電解副分類800'は「定格電圧」と呼ぶ属性827を有し、一つの事例871は5Vというパラメータ890とタイプBケースのパラメータ888を持つ。ほとんどのオブジェクトやリストは説明を簡単にするために不完全にしか示していない。また図41から50の説明では同じ対象物を参照するのに同じような参照値を用いている。

図51で分類オブジェクト800は名前806を有し、ここでは「エレクトロニクス」である。分類オブジェクト800は副分類リスト893を指すフィールド802を持つ。リスト893は副分類800'のハンドルである第1のエントリー894を持つ。この場合副分類800'の名前806'はキャパシターである。もちろん枠組みオブジェクトを参照するために図51には示していないが、ハンドルを使い、ハンドルマネジャー137とハンドル表を使う。これらは図を簡単にするために図51には示していない。

副分類800'キャパシターは副分類893'のリストを指すフィールド802'を有する。リスト893'はエントリー894'を持ち、それは副分類800'のハンドルである。副分類800'の名前806'は電解である。副分類800'はフィールド802'に空のエントリーを持つ。それは通常は副分類リストへのポインターとなるものである。ここでは、副分類800'はさらなる副分類を持っていない。

キャパシター副分類800'に戻って、フィールド803'は属性リスト897へのポインターを含んでいる。リスト897は「ケースタイプ」と呼ぶ列挙した属性827へのハンドルを有している。列挙した属性オブジェクト827のフィールド830は、「キャパシター」と呼ぶ副分類800'を定義するハンドルを持っている。列挙した属性オブジェクト827はポインター835を持ち、それはエニユマレーターへのハンドルリスト839を指し示す。ここでは、リスト839はエニユマレーター841へのハンドル898を持っている。エニユマレーター841はこのエニユマレーターのための外部名のポインター842を持つ。

外部名は「ケースA」というASCII列である。同様に、リスト839の項899はケースBに関してエニユマレータ841'を指し示す。電解と名付けられた副分類800'に戻って、ポインター803'は属性リスト897'を指し示し、リスト897'の一つのフィールドは数字属性827'「定格電圧」へのハンドルを持っている。数字属性827'はフィールド830'を有し、それは分類を定義するハンドルを持っている。この例ではそれは電解と名付けられた分類800'である。数字属性オブジェクト827'はフィールド838'を有し、それは電圧単位ファミリー（図示せず）のハンドルを持っている。電解副分類800'に戻って、フィールド804'は、事例のハンドルリスト895へのポインターを持っている。リスト895の項896は事例871に関するハンドルを持っている。事例871はその分類のハンドルを有するフィールド873を有している。ここでは電解副分類800'である。また事例データオブジェクトはパラメータリスト877を指し示すフィールド875を持っている。リスト877は数字パラメータ890を指し示すポインター878を持っている。数字パラメータ890は、属性827'（定格電圧）のハンドルを持ったフィールド881を有している。数字パラメータオブジェクト890はまた、フィールド884を有し、それは単位のハンドルである。ここではそれは「ボルト」である。簡単のために、単位オブジェクトは示していない。数字パラメータオブジェクト890は値5.0を持ったフィールド885を有している。この例では、電解キャパシタは5.0ボルトの定格である。

パラメータリスト877は列挙したパラメータ888を指し示すポインターを持っている。列挙したパラメータオブジェクト888はフィールド881'を持ち、それは属性のハンドルである。ここではそれはケースタイプである。列挙したパラメータオブジェクト888はまたエニユマレータ841'へのハンドルとなるフィールド882を有している。この例では、定格5.0ボルトの電解キャパシタはタイプケースBである。

ここで述べたデータ構造は大きな利点がある。図51を参照して、このデータ構造の名前や記述を変えることは容易である。タイプBケースの1000例のキャパシタを持ったデータベースを例として考えよう。タイプBケースが途絶え

たりあるいは「再生品」という名に変わったら、必要な変更はそのケースタイプを示しているASCII列を一つ取り替えるだけでよい。データベースの中の1000例すべてにハンドルが付いていて、ハンドルマネジャー137がそれをASCIIテキスト列と関連づけてくれる。データベースへの他の変更は行う必要がない。

本発明のデータ構造の他の利点は、最初の値が決まっていなかったときにはなにも格納しないことである。このことで、無駄なスペースが除かれる。

データ構造の他の利点は、ツリー構造の場所によってアルゴリズムを変える必要のないことである。ツリー構造の場所に関係なくすべてのアルゴリズムが同じように働く。特別なケースはルート分類だけである。例えば、データベースに事例を追加するアルゴリズムはツリー構造のどこであるかに関係なく同じである。これは枠組みに大きな変更を加えるのがきわめて容易である。ツリー構造のある分類あるいは枝全体を一つの場所から他のところに移動させるのに、ハンドルのリストを単に要えるだけで行える。変換プログラムを走らせる必要がない。分類オブジェクト800はその親801のハンドルを持っているので、誰がその親なのかを知っている。また、分類オブジェクト800はその副分類のリストへのポインター802を持っているので、誰がその子供であるかを知っている。このデータベース構造では、事例を削除することが早くできる。事例リスト804の最後の項を取って、それを削除事例の場所に移すことで事例を削除できる。他の言い方をすれば、最後の事例のハンドルを削除事例のハンドルに重ねて書くと、リストの項目数が一つ減る。事例オブジェクト871のための事例インデックスフィールド874を使うと早く削除するのに便利である。

望ましい実施態様において、パラメータの値はいつもベース単位に格納されている。フィールドのオブジェクトは一語文のメモリーを占める必要はない。特定のパラメータはすべて隣り合って格納される。これはサーチの早さを上げる。例えば、図51を参照して説明したケースタイプ841'は、ケースタイプの他のパラメータと隣り合って格納される。5.0ボルトの数字パラメータはメモリーの中で他の数字のボルトパラメータと隣り合った物理的な場所に格納される。上で述べたように、その分類にサブツリー事例カウントをするフィールド808

を分類オブジェクト構造800に与えているので、システムは部品カウント172をすぐに表示するのでユーザーがツリーを通して検索している間にすぐにフィードバックをすることができる。ある部品を見つけるプロセスは、必要な属性を持たない何千もの部品を処分して必要な属性を持った数少ないものに検索を絞り込むことと基本的には同じである。

これは分類階層のルートを正しい分類にたどって行って成し遂げられる。このフェーズを通して、サブツリー事例カウントを示しているデータ構造フィールド808を使って、見つけた部品の指標172は更新される。このことによって各ステップで使える部品を実際にカウントするのに比して、応答時間をきわめて短くできるという大きな利点がある。ユーザーは、選択したツリーで使える部品の数を調べるためにすぐにフィードバックができる。そのときの検索条件や分類についての事例の数をすぐにフィードバックすることができるので、オブジェクトオリエンテッドな階層ツリー構造と必要な属性を組み合わせた検索条件によって、従来から試みられていたデータベースマネジメントスキームに比して、大きな利点となった。

ダイナミッククラスマネジャー134の重要な働きは、操作の間ずっとデータベース構造を更新できると言うことである。データベース構造は枠組みとして知られている。オブジェクトオリエンテッドなデータベースの枠組みは分類を使うように構成されている。分類は属性を含む。属性はエニユマレータや単位ファミリーを含むことができる。これらの項目を付け加え、移動したり、削除したりできる能力は、データベースのダイナミックオペレーションのために重要なことである。

枠組みに分類を付け加えるには、3つの項目を知る必要がある。分類名と、新しい分類の親と、新しい分類を入れる副分類リストの中の場所とである。図65はこの操作を示している。最初のステップ1840では、親分類のハンドルを実際の分類ポインターに変換する。親ポインターは使う前にステップ1841ですぐにテストされる。そのポインターが有効なものでないことがわかったら、操作はステップ1842で停止する。そのポインターが有効なものであれば、挿入インデックスがステップ1843でテストされる。それが有効でないことがわかっ

たら、操作はステップ1844で停止する。結局、分類名がステップ1845でテストされて、それが分類名のガイドラインに沿った有効なものかどうかを決める。分類名がよくなかったら、操作はステップ1846で止まる。ステップ1845で分類名を受け付けいたら、新しい分類が作られる。新しいハンドルがステップ1847で最初に作られて、それから新しい分類がステップ1848で内部メモリーに作られる。新しいハンドルは図66のステップ1849で分類表に挿入されて、ステップ1850で副分類の親リストにそのハンドルが追加される。最後の操作は第2の格納装置103の親を示しているところに新しい分類を付け加えることをファイルマネジャー140にさせることである。

分類に属性を付け加えるには、3つの項目を知る必要がある。それを持っている分類の分類ハンドルと、新しい属性を挿入する場所と、属性の名前である。図67は属性を付け加えることを示している。最初のステップ1930は、分類ハンドルを分類ポインターに変換することで、そして、それが有効な分類ポインターかどうかを1931でその分類ポインターをテストする。そうでなければ、手順は1932で止まる。分類ポインターが有効であるとなったら、挿入インデックスの有効性を1933でテストする。インデックスが有効テストでだめになれば、手順は1934で止まる。インデックスが有効なものであれば、操作は1935に続き、属性名がテストされる。属性名がよくなければ、操作は1936で止まる。列挙した属性名が1935で受け入れられたら、その属性が作られる。ステップ1937でその属性への新しいハンドルが作られる。そして新しい属性がステップ1938で作られる。1939でそれを持っている分類の場所の属性リストに新しい属性ハンドルが追加される。最後のステップは図68の1940で、ファイルマネジャー140で第2の格納装置103を更新して新しい属性を入れる。その操作はステップ1941で完成する。

事例の追加は図69に示している。事例を追加するには分類ハンドルが必要である。分類ハンドルは1918で分類ポインターに変換される。1919で分類ポインターは有効な分類ポインターかどうかをテストされる。それが有効なものでなければ、操作は1920で止まる。分類ポインターが有効なものであるとなったら、操作は続いて1921で新しい事例ハンドルと新しい事例オブジェクト

が作り出される。新しい事例のハンドルは1922でハンドル表に挿入される。事例が1923で親の事例リストに加えられる。サブツリー事例カウントは1924で新しい事例を反映して増える。その事例はメモリーに作られて第2の格納装置103に加えられる必要がある。それは図70のステップ1925で行われる。操作はステップ1926で完成する。

分類の削除は図71に示す。データベース構造から分類を除くには、現行の分類ハンドルを特定しなければならない。分類ハンドルは最初にステップ2600で分類ポインターに変える。分類ポインターはそれから有効な分類ポインターかどうかを2601でチェックされる。その分類ポインターが有効なものでなければ、操作は2602で止まる。分類ポインターが有効なものであれば、それがルート分類かどうかを2603でチェックされる。その分類ポインターがルート分類のものであれば、手続きは2604で止まる。というのは、ルート分類は削除できないものであるから。その分類ポインターがルート分類を示していないものであれば、その分類がリーフ分類かどうかを2605でチェックされる。その分類ポインターがリーフ分類のものであれば手続きは2604で止まる。その分類ポインターがリーフ分類を指し示していることがわかれば、操作は続いて、2906でその分類の事例すべてが削除される。事例を削除するプロセスは図75を参照して下に述べる。ステップ2607で削除される分類にあるすべての属性が削除される。図72のステップ2608で、その分類のその親分類とのリンクをなくす。システムはリンクをうまくはずしたかどうかをチェックし、またその分類リストを持つデータ構造が完全なままかどうかを2609でチェックする。リンクをはずすのに成功した場合、操作は続き2611で分類オブジェクトは実際に削除される。ステップ2612で、第2の格納装置103から分類オブジェクトを取り除くようにファイルマネジャー140は指示を受けて、操作はステップ2613で完成する。

属性の削除は図73に示す。属性を削除するには、ステップ1860で属性ハンドルは属性ポインターに変換されなければならない。ステップ1861はステップ1860で得た属性ポインターが有効なものであるかどうかをチェックする。属性ポインターが有効なものでない場合には、操作は1862で止まる。属性

ポインターが有効であれば手続きは続いてステップ1863で、その属性から派生しているサブツリー事例のすべてのパラメータをそのサブツリー全体にわたって検索する。検索の後、ステップ1864で、その属性から何個のパラメータが派生しているかをシステムが決める。この属性から派生しているパラメータがあれば、操作は1865に進み、そこでパラメータは定義なしとなる。その属性から派生しているパラメータがない場合には、操作はステップ1866に続く。同様に、1865でパラメータが定義なしとなった後も、操作は1866に続く。ステップ1866で、属性は定義をしている分類からのリンクをはずされる。リンクをはずす操作が成功したかどうかを1867でチェックする。リンクをはずすのに失敗したら、操作は1868で止まる。リンクをはずすことが成功であれば、属性オブジェクトは図74の1869で削除される。ステップ1870でその属性を第2の格納装置103から取り除くようにファイルマネジャー140に指示が行く。操作はステップ1871で完成する。

事例の削除は図75に示す。事例をデータベースから削除するには最初にする2000で事例ハンドルを事例ポインターに変換する。2001で、その事例ポインターが本当に有効なものであったかをチェックされる。その事例ポインターが有効なもの でなければ、操作は2002で止まる。その事例ポインターが有効なものであれば2003でその事例はそれを持っている分類からのリンクをはずされる。事例オブジェクト自体は2004で削除される。一つの事例がサブツリーから削除されたことでサブツリー事例カウントは2005で減少する。2006でファイルマネジャー140は、事例の削除を反映して格納103を更新するよう指示を受ける。その操作はステップ2007で完成する。図76に、サブツリーを分類階層の新しい場所に移動させることが書いてある。ステップ1800で、サブツリーを移動させる手続きには、動かす分類と、行き先の分類と、特定した行き先のきょうだい分類の位置関係が必要である。ステップ1801で、動かす分類の分類ポインターと行き先の親の分類が得られる。ステップ1802ですべてのポインターが有効かどうかをテストできないときには、ステップ1804でエラーとなる。そうでないときにはテスト1805が行われて、その分類をそれ自体の親から小さな移動をさせないようにする。ステップ1806で行

き先の親の分類の副分類間の位置が有効な範囲になるようにして、誤った場合ステップ1804に戻ってエラーとなる。ステップ1807で移動させる分類と行き先の分類の両方の分類階層を調べて最も近い共通の祖先の分類を特定する。図77のステップ1808で共通の祖先を調べて、それが動かしている分類と同じかどうかを決める。もしもそうであればその分類をその親に移動させていないということを確認するテストをすでに行っているのであれば、これは分類をその副分類へ移動させようとする試みであることになり、エラーで戻る。他のすべての移動は正当なものであり、ステップ1809で分類をその親分類からはずして、ステップ1810で行き先の副分類リストに加えられる。ステップ1811で、行き先の分類サブツリー事例カウントは、動いた分類にある事例の数だけ増やされる。そして、動いた分類が元にあった親分類のサブツリーカウントはステップ1812で動いた分類事例カウントだけ減らされる。ステップ1813で、知識ベースのパーマネントイメージはファイルマネジャー140を通して更新されて、ステップ1814で成功裏に指示者に戻る。

図78は移動させた分類をその元の親分類からはずすことを説明している。ステップ1815で親の分類ポインターが得られて、ステップ1816で親分類の副分類リストを得るのに使われる。ステップ1817でテストされて移動させる分類の分類が得られた副分類リストにないことがわかれば知識ベースは内部的に一致していないとしてエラーとなる。そうでなければステップ1818でその分類は親の分類の副分類リストから削除されてステップ1819で成功となる。図79は移動させる分類に最も近い共通の親と行き先の分類を見つけるプロセスを説明している。ステップ1820で移動させる分類のハンドルとして仮の分類ハンドルがセットされる。ステップ1821で仮の分類の親を得て、その分類をルートに移すために分類表を作り出すループが開始される。ステップ1822で出会った分類をリストに追加して、ステップ1823でルートに出会ったらその繰り返しが終わる。ステップ1823でルートに出会わなかったらステップ1824で仮の分類ハンドルをその親分類のハンドルにセットしてその繰り返しを継続する。

図80に行って、ステップ1831から1828で、行き先の分類と同様なり

ストが作られる。ステップ1831で、行き先の分類ハンドルとして仮の分類ハンドルがセットされる。ステップ1832で仮分類の親を得て、その分類をルートに移動させるために分類リストを作るループが開始される。出会った分類をステップ1826でリストに追加して、ステップ1827でルートに出会ったらその繰り返しが終わる。ステップ1827でルートに出会わなかったら、仮の分類ハンドルをステップ1828で親分類のハンドルにセットして繰り返しを継続する。

最後のステップ1829は、2つの得られたリストで合致した分類ハンドルが見つかるまで繰り返される。これは、最も近い共通の祖先のハンドルであり、そしてステップ1830に戻る。

2. コネクティングマネジャー

コネクティングマネジャー135は知識ベースサーバー132のサブシステムであり、現在のクライアントコネクションに関する情報を扱う。コネクティングマネジャー135はクライアント130、133、144の知識ベースサーバー132へのコネクションをさせたり、維持したり、閉じたりする働きがある。コネクティングマネジャー135は各クライアント130、133、144のコネクションについてクエリーマネジャー136の事例を作り出す。コネクティングマネジャー135はこれらのクライアントコネクションへのエントリーのリンクしたリストを維持する。コネクションマネジャーが持っているデータを図で示したものが図81である。

図81で、コネクションマネジャー135はコネクションリストポインター1070を持って、コネクションリスト1077を指示し示す。コネクションリスト1077は、クライアント130、133、144の開始時間、最後の要求のあった時間、最終メッセージ1071の時間についてのデータを持つ。API143をコールした全カウント1072も持つ。リモートプロセスジャーコールコネクション情報1073のポインターも持つ。関連したデータベースマネジャー139に関する情報のポインターも持つ。コネクションマネジャー135はアクセスを制御する読み出し専用フラグ1075を持ち、また関係するクエリーマネジャー136のポインターも持つ。

3. クエリーマネジャー

クエリーマネジャー 136 は知識ベースサーバー 132 のサブシステムであり、ダイナミッククラスマネジャー 134 とともに知識ベース 123 にクエリー操作を行う。クエリーマネジャー 136 はクエリーデータベース構造を管理し、選択子をパラメータに一致させたり、クエリーにあった事例や分類のリストを作り出す働きがある。

次の説明は図 158～図 163 に書いたデータ構造を参照するものである。クエリーマネジャー 136 が事例を挙げたときに、クエリーマネジャー分類 700 が作られる。この分類 700 の各事例は、クエリーについてのクエリーハンドルマネジャーと、クエリー結果 712 についてのクエリーハンドルマネジャーと、検索結果 713 についてのクエリーハンドルマネジャーとを含む。一般に、クエリーハンドルマネジャー分類 701 はベースクエリー分類 702 のリストである。このリスト 701 はハンドルとベースクエリー分類 702、すなわち派生分類、クエリー分類 703、検索結果分類 704、クエリー結果分類 705 の一つとの間の関係である。リストへのオフセットはオブジェクトのハンドルを示す。

「クエリー」とは API 143 で作られるオブジェクトであり、パラメトリッククワイテリオンをもとにして知識ベース 123 にある事例を選択するのに用いられる。クエリーはそれが作られるといつも分類と結びついている。

クエリーを作るためには、ベースクエリー分類 702 の派生分類としてクエリー分類 703 が作られる。作られたすべてのクエリーにはこれらの分類 703 の一つが付いている。ベースクエリー分類 702 は、クエリーのベース分類（クエリー分類 703）、クエリー結果（クエリー結果分類 705）及び検索結果（検索結果分類 704）である。ベースクエリー 702 にはクエリー分類ハンドル 714 があり、それはその上でクエリーが作られた分類ハンドルである。クエリーマネジャー 136 はダイナミッククラスマネジャー 134 にアクセスする必要があるので、ダイナミッククラスマネジャー 134 へのレファランス 715 が保持されている。

一度作られると、クエリー分類 702 は外部から削除されるまで存続し続ける。

クエリーは零かそれ以上の「選択子」を持つ。「クエリー選択子」はそのクエリーが作られた分類で決まる属性の一つと関連する。選択子をセットすることで、その選択子にあった事例にクエリーを戻す。多くの選択子をセットすることは、その選択子にあった事例の結びついたものにクエリーを戻すことになる。すなわち、すべての選択子にあった事例のみに戻す。

クエリー属性分類 706 の選択子の正確な形は、関係する属性のタイプによって決まる。各属性はそのクエリーに対して、多くとも一つの関連する選択子分類 706 すなわち派生分類 707, 708, 709, 710 を持つ。いかなる属性タイプについても、選択子分類 706 すなわち派生分類 707, 708, 709, 710 は、関係するパラメータが「定義なし」状態 731 であるような事例を含むようにセットできる。定義なしのパラメータを必要としないように選択子分類 706 をセットすることは、定義なしとセットされたパラメータをもった事例を除くことになる。選択子分類 706 をそうでないようにセットしないで定義なしのパラメータを含ませようとすることは、パラメータが定義されていない事例のみを生じることになる。

API 143 コールが特定の属性選択子分類 706 をセットした結果として、属性選択子分類 707, 708, 709, 710 がクエリー分類 702 属性選択子分類リスト 716 に加えられる。

属性選択子分類リスト 716 は、クエリー分類 702 が壊されると壊される。

クエリーを実行することを「クエリーを適用する」と呼ぶ。クエリーを適用することはクエリー結果ハンドルの行う。クエリー結果ハンドルはクエリー結果分類 705 を参照する。クエリー結果分類 705 は、クエリーが行われた事例リスト 723 を含むオブジェクトである。クエリー結果ハンドルを与えて、ユーザーはリスト 723 で示されている事例を検索することができる。クエリーの結果は外部から削除されるまで存続し続ける。クエリー分類 702 を後で変えてもそのときにあるクエリー結果分類 705 に影響を与えない。クエリー分類をその後で適用することは他のクエリー結果分類 705 となる。

クエリーはローカルにあるいはサブツリーに適用することができる。クエリーをローカルに適用した場合、クエリーマネジャー 136 は、クラスマネジャー 1

34のレファランス715を用いて、分類ハンドル714に関係する分類ポインターを検索する。分類ポインターに関係する事例リスト検索されて、リストイタレータを用いて、そのクエリー分類703の中の属性選択子リスト716に対して各事例を評価する。選択子のないクエリー分類703は分類714のすべての事例を単に戻すだけである。

図82と図83では、ローカルクエリーを適用している。クエリーハンドルはステップ750でクエリー分類702ポインターに変換される。ステップ751でポインターの有効性を見る。ポインターが無効であればエラーとしてステップ752に戻る。

ステップ753で、クエリー結果分類705が作られてクエリー結果ハンドルマネジャー712に追加される。

ステップ754でクラスマネジャー134が呼ばれて、レファランス715を用いてクエリー分類702にセットされている分類ハンドル714への分類ポインタを得る。このポインターはステップ755でクラスマネジャー134ファンクションを呼んで714で分類ハンドルの事例リストを得るのに用いられる。ステップ750からクエリー分類702の分類ポインターが用いられて、ステップ756でクエリー分類702に関係する選択子分類リスト716を検索する。

本発明を最も効果的に実行することは、なにか選択子分類706および派生分類がセットされているかどうかをステップ757でチェックすることである。選択子分類706や派生分類がセットされていないならば、ステップ760が実行されて分類事例リストをクエリー結果分類705に関連づけて、通常のリターンがステップ761で行われる。

選択子706と派生分類がセットされているのならば、ステップ758で調べられる分類事例リストに事例が必要になる。ステップ759でそれ以上の事例がないならば、合致した事例リストをステップ760でリスト723のクエリー結果分類705と関連づけて、ステップ731で通常のリターンが行われる。選択子分類706と派生分類がステップ762でクエリー分類702の属性選択子分類リスト716から検索される。ステップ763で、評価すべき他の選択子分類706や派生分類がなければ、ステップ769で事例ハンドルをクエリー結果分類

705の事例リスト716に保存した後、ステップ758に進みその分類の次の事例を得る。

評価すべき選択子分類706やその派生分類の一つがあれば、ステップ764が実行される。このステップで属性ハンドル763のパラメータ値を検索する。

本発明を最も効果的に実行する他のことは、ステップ765で開始するプロセスで、パラメータが定義されているかどうかをチェックすることである。このステップは空か零の値を取り扱うのに効果がある。パラメータが定義されていないならば、定義なしの選択子フラグ731がステップ768でチェックされる。定義なしのフラグ731がセットされていない場合、その事例ハンドルは合致の可能性があると見て捨てられて、次の事例がステップ758で処理され始める。定義なしのフラグ731がセットされておれば、事例は合致し、次の選択子リスト716の項目がステップ762で処理される。

パラメータ値がステップ765で存在するならば、選択子リスト716の項目がステップ766でチェックされて条件がセットされているかどうかを見る。そうでなければ、次の事例がステップ758で処理される。属性タイプをもとにして、選択子706と派生分類が使われてステップ767でパラメータと選択条件を評価する。選択条件が合致すれば、ステップ762で次の選択子分類リスト716の項目が処理される。

図84に、サブツリーについてクエリーを実行するプロセスを示している。ステップ770でクエリーハンドルはクエリー分類702のポインターに変換される。771で、ポインターの有効性を見る。ポインターが無効であれば、ステップ772でエラーとして戻る。

ステップ773で、クエリー結果分類705が作られてクエリーハンドルマネジャー712に追加される。次のステップ774で図82と図83に示したローカルクエリー機能の適用を実行する。このステップ774は分類ハンドル714の各副分類について繰り返して実行される。この分類ハンドル714の副分類はステップ775でクラスマネジャー134のレファレンス715から検索される。

。

ステップ776で処理すべき他の副分類があれば、その副分類の分類ハンドル

は、ステップ781でクラスマネジャー134のレファレンス715から検索される。図82と図83のローカルクエリー処理はステップ774から入る。ステップ776から、もしもすべての副分類が処理されていたら、操作はステップ777で親分類に戻る。ステップ778で繰り返しのアルゴリズムがクエリー分類714のトップに戻っているかどうかをチェックする。そうでなければ、ステップ776で現在の分類の残りの副分類について処理をする。クエリー分類714のトップに戻っておれば、ステップ779で事例リスト723をクエリー結果分類705と関連づけて、ステップ780で操作は要求者に戻る。クエリーがつけられ適用したときに作られたデータ構造は、図164、図157、図156および図155に示している。これらの図では、特定な知識ベースのルート分類はボーレ属性「不連続」を持ち、「インチ」のベース単位をもった数字属性「長さ」を持っていると仮定している。ユーザーは3インチを超えた長さをもったすべての不連続な部品にクエリーを適用する。

図164で、ルート分類（分類ハンドル0）についてのクエリーが、API143の機能「pmx createquery」（ピーエムエクス クリエイトクエリー）を使って作られている。2つのクエリーハンドルは要求者に戻る。ユーザーは「pmx getattributedescriptorset」（ピーエムエクス ゲットアトリビュートデスクリプション）を得るAPI143の機能を用いて、ボーレ属性「不連続」（属性ハンドル10）と数字属性「長さ」（属性ハンドル19と単位ハンドル5）についてのハンドルを検索する。

図157で、クエリーハンドル2とAPI143の機能「pmx setbooleanselector」（ピーエムエクス セットブーレンセクタ）を使っているボーレ選択子のセット結果を示している。

図156で、クエリーハンドル2とAPI143の機能「pmx setnumericselector」（ピーエムエクス セットニューメリックセクタ）を使っている数字選択子のセット結果を示している。

クエリーハンドル2を適用した後で、事例3, 300及び3000が見つかる。クエリーを適用した結果を図155に示している。この例では他のクエリー結果がないのでクエリー結果ハンドル0に戻る。

本発明をクエリ (query) カウントに適用した別の重要な最適実施例が第 85 図で説明されている。この方法は、スキーマクラスツリーで利用できる部品番号を高速に効果的にリトリーブ 130 に戻すのに使用される。クエリクラス 702 のポインタがクエリハンドルにより変換される時、ステップ 790 で処理が開始される。ステップ 791 はこのポインタの有効性をチェックし、そしてもしエラーが発生したら、ステップ 792 に戻る。ステップ 793 は、クエリクラス 714 に対するクラスポインタを獲得するために参照 715 を使用してダイナミッククラスマネージャ 134 にアクセスする。ベースアトリビュートセレクトククラス 706 により記述されるセレクト 716 のリストはステップ 794 で検索 (リトリーブ) される。もし、選択リスト 716 のアイテムがステップ 795 で決定されるとして存在するなら、第 84 図に記載されたクエリを適用する手続きが実行されなければならない、そしてインスタンスカウントの結果がステップ 798 で戻される。重要な発明がステップ 796 にある。ダイナミッククラスマネージャ 134 は、そのクラスが親であるサブツリーの全インスタンスのカウントと同様に、ローカルインスタンスのカウントを直接維持する。この値は、インスタンスが移動した時に維持され、削除され、もしくはクラスに付加される。クエリ適用カウント手続きが適用される時、クラスにおいて値が単純に参照され、そして 796 でユーザに返される。このステップは、高性能にリトリーブ 130、アイテム 172 にツリートラバーサルフィードバックする。

クエリの結果 705 におけるインスタンスの初期オーダはランダムである。クエリの結果のインスタンス 723 はインスタンスをソートするための A P 143 機能を使用して再オーダできる。リトリーブ (検索) 機能は、ソートされたオーダのインスタンスに戻る。クエリ結果クラス 705 は、多数回再ソート可能である。

第 161 図において、ソート要求は、アトリビュート 719 の順番化されたリストおよび順番がソートオーダリスト 720 において昇順であるか、あるいは降順であるかどうかの指示により構成される。降順は、昇順である通常の順番の全くの逆である。

インスタンスは、オーダリングリスト 719 の第 1 アトリビュートにより最

初にオーダされる。第1アトリビュートと等しいセッティングをもつインスタンスグループの中で、第2アトリビュートが使用され、そして、リストが消滅するまで同様である。リストにより一意的に決定されていないオーダは、本質的にランダムである。

ブーリン (Boolean) アトリビュートに対する昇順オーダは、(真, 偽, 未定義) である。計算されるアトリビュートに対する昇順オーダは、未定義で続くスキーマで定義されるような演算子 (エニューメレータ) のオーダである。ストリングアトリビュートに対する昇順は、未定義で続く通常ASCII対照シーケンスである。ニューメリックアトリビュートに対する昇順オーダは、スキーマにより定義されるオーダのベースユニットにより最初にソートされる。ベースユニットの中で、インスタンスはニューメリックシーケンスの中にある。ニューメリックアトリビュートに対する昇順は、スキーマにより定義されるオーダのベースユニットにより最初ソートされる。ベースユニットの中で、インスタンスはニューメリックシーケンスである。未定義のパラメータは最後である。

ファイルマネージャ 140 から派生する付加が可能である。ファイルマネージャ 140 によりダイナミッククラスマネージャ 134 およびハンドルマネージャ 137 に与えられるインタフェースは第2永久記憶103のダイナミッククラスマネージャスキーマおよびインスタンスデータのコピーを保持することに同意する。スキーマに変更がなされるに従って、変更がまた第2記憶になされる。ダイナミッククラスマネージャ 134 は、ファイルマネージャ 140 を介して第2記憶103からデータを読むことにより初期化される。インタフェース仕様に従う他の第2記憶機構が実現できる。他の実施例は、インフォーマックス (Informix) データベース、オラクル (Oracle) データベース、レーマ (Reima) データベース等のリレーショナルデータ管理システムを含む商用データベースを使用できる。他の実施例は他の固有のフォーマットを使用して構築できる。

クエリの結果は、インスタンスハンドルで具体化され、そして移植される。インスタンスのオーダは、クエリの結果の申でランダムである。クエリの結果の中のインスタンスは、特徴基準に従ってソートされる。複数のソート基準が特徴化

できる。ソート基準は、一つもしくはそれ以上のアトリビュートのリストおよび各アトリビュートに対するソートオーダ（昇順もしくは降順）を示すことにより特徴化される。クエリの結果のインスタンスは、特徴化されたオーダに従うアトリビュートにより指示されるパラメータをソートすることによりオーダされる。

クエリの結果は、ソート基準を含むソートメッセージを受け取ることに基づいて即座にはソートされない。むしろ、クエリの結果は、単にソート基準を記憶しているだけであり、インスタンスハンドルがクエリの結果を要求される時にソートを実行するのを待つ。（インスタンスハンドルに対する要求はRMXリトリバに顕著である）。

インスタンスハンドルがクエリの結果を要求される時、クエリの結果は、この時点で自身をソートするであろう。しかし、完全なクエリの結果はソートされない。対象のインスタンスを含むクエリの結果の一部分のみが単にソートされる。使用されるソート方法は増加的であり、対象のインスタンスを含むクエリの結果のそれらの部分を単にソートするだけである。クエリの結果の他の部分は、ソートされないで残される。増加的ソートは、応答時間を改良するためになされる。クエリの結果の一部分をソートすることは、通常、クエリの結果全体をソートするより少ない時間である。

インクリメンタルソートは、クエリの結果においてインスタンスハンドルがソートされたおよびソートされないトラッキングを必要とする。これを達成するために、クエリの結果が領域にサブ分割される。領域には二つのタイプがある。クエリの結果の各インスタンスハンドルは、ソートされたもしくはソートされないでいずれかにある。クエリの結果の各インスタンスハンドルはソートされたもしくははされない領域にある。領域マネージャはそれらの領域を管理する責任を負う。

最初、ソートに先立って、全体的クエリの結果は、単一の未ソート領域に含まれる。増加的にソートする動作は、クエリの結果をいくつかの領域、あるものはソートされ、あるものはソートされない、にサブ分割する。クエリの結果の部分がソートされればされる程、ソートされない領域数は少なくなる。結局は、全体のクエリが単一のソートされた領域になる。

領域マネージャは、ソートされたおよびソートされない領域についての意味を意識し、実行速度を遅らせる領域の細分化を避けるために領域をまとめて結合するのにその情報を使用する。領域マネージャは、領域をまとめて結合するのに二つの結果を使用する。即ち、1) 同タイプ（ソートされたもしくはソートされない）の二つの隣接領域はより大きい単一の領域にでき、2) 単一のインスタンスハンドルのみを含むソートされていない領域は、より大きいソート領域にするように隣接ソート領域と結合される。

ソートに先立ち、クエリの結果は、主に、第144図のソートされていない状態のステップ1350で0からNのインスタンスのリストから構成される。ソートメッセージの受け取りは、クエリの結果の状態をソートされた状態に変更し、そしてそれを第144図のソート基準ステップ1351に与える。

クエリの結果は、最終的には、第188図のステップ1352のインスタンスのハンドルのリストから、1番目のインスタンスのハンドル戻すためのメッセージを受け取る。この点において、クエリの結果は、1番目のインスタンスを正確にオーダーするためにリストをソートしなければならない。これは、0（領域の低端）とN（領域の高端）の間のランダムインデックスRを最初に選択することによりなされる（ステップ1253）。全体のリストは、Rにおけるインスタンスより大きい全インスタンスがリストにおいてRより上に置き換えられ、そしてRにおいてインスタンスより小さいインスタンスはリストにおいてRより下に置き換えられる（ステップ1354）。Rのインスタンスは、ここでリストの中で普通にソートされる。この点で、3つの領域がある。即ち、0からR-1（ソートされていない）、R（ソートされている）、そしてR+1からN（ソートされていない）。

もし、1番目とRが一致すれば、ソートは完全であり、そして1番目のインスタンスハンドルが戻される。そうでなければ、ソートが続けられねばならない。ソートはRとの関係において1番目の位置にベースを置いて続けられる。1番目がRより小さければ、その時、0（低端）とR-1（高端）の間の領域がオーダーされる。もし、1番目がRより大きければ、その時、R+1（低端）とN（上端）の間の領域がオーダーされる。決定された適切な領域により、新しい領域R

が新しい領域の中に選択され、そして新しい領域の全インスタンスが部分的にRにおけるインスタンスに関して部分的にオーダーされる。

リストは、領域に繰り返しサブ分割され、そして部分的に1番目とRが一致するまでオーダーされ、その時点で、ソートは中断し、1番目のインスタンスハンドルが戻される。

他の1番目のインスタンスが要求された後で、この1番目のインスタンスがまだソートされていない領域の中に見出されたら、ソートされる必要はなく、そして1番目のインスタンスハンドルは、すぐには戻されない。もし、この1番目のインスタンスがソートされた領域の中に見出されないなら、ソートは1番目が見つかるまで領域を繰り返しオーダーすることにより続く。

各回に、領域がオーダーされ、新しい領域が識別され、領域マネージャがこの情報に与えられる。この領域マネージャは、あらゆる領域の軌跡を保持する。各繰り返しの開始において、領域マネージャは1番目が記憶領域に存在するかどうかを訊ねられる。もし、そうであるなら、その時、さらにソートは必要とされない。もし、1番目がソートされた領域に見出されなければ、その時、領域マネージャは、1番目が存在する領域の境界の下端および上端を与え、そしてアルゴリズムはその領域をオーダーしそして新しい領域を生成する。

ソートセッションの終わりに、1番目のインスタンスに戻る直前に、領域マネージャは、前に説明した規則に基づいて細分化された領域を結合する機会を与えられる。ソートセッションの終わりに、領域マネージャは、第189図の1355に示されるステップをトラッキングする。同じタイプの任意の隣接する領域がまとめて結合される。また、サイズ1のソートされていない領域がソート領域の隣になるように結合される。

その領域マネージャは、領域結合が完全である時、1356で示される領域をトラッキングする。

クラスパターンサーチは、クエリがサブツリーにおけるクラスの全部に適用される以外は、クエリと同様である。サーチ結果は、サーチ結果クラス704により表される。

クラスパターンサーチはクラスネームのパターン一致を実行し、名前がパター

ンに一致するクラスのリストに戻る。クエリと違って、サーチ生成とその実行を分離することはない、というのは、サーチは、一度に断片をまとめて上げる必要がある程複雑ではない。サーチは、ユーザがその配置をはずすまで存在し続けるアイテムである“サーチ結果”ハンドルに戻る。クラスが戻される定義されたオーダーはない。

4. ハンドルマネージャ

ハンドルマネージャ 137 は、全オブジェクトに対する生成、削除およびハンドルのディスク対メモリマッピングサービスを提供するダイナミッククラスマネージャ 134 の部品である。ハンドルマネージャ 137 は、第 42 図に示される仮想メモリアドレスの 2 つのリストにより構成される。第 1 リスト 810 はスキーマオブジェクト（クラス、アトリビュート、演算子、ユニットおよびユニットファミリー）の仮想メモリアドレス 810-814 を含む。第 2 リスト 811 はインスタンスの仮想メモリアドレス 815-826 を含む。ハンドルはリストへのインデックスである。このように、スキーマオブジェクトハンドルもしくはインスタンスハンドルが与えられると、ハンドルマネージャ 137 は、ダイナミッククラスマネージャ 134 に望みのオブジェクトの仮想メモリアドレスを戻す。

ダイナミッククラスマネージャ 134 がそれをもつハンドルに対してあるオブジェクトに対するデータを確かめる必要がある時、ハンドルマネージャ 137 は第 52 図に示されるようにオブジェクトの仮想メモリアドレスに対する要求に答える。手続きは、ダイナミッククラスマネージャ 134 からの要求をもつステップ 1000 で始まる。ハンドルは、ステップ 1001 で有効性をチェックされる（例えば、ハンドルがハンドルマネージャ 137 により生成されるもの）。もし、ハンドルが有効でなければ、エラー条件が生成され、そしてハンドルマネージャは、ステップ 1002 でエラーを指示するためのダイナミッククラスマネージャ 134 に NULL 仮想メモリアドレスを戻す。そうでなければ、ハンドルマネージャ 137 はステップ 1003 で続く。

ハンドルが有効であれば、その時、適当なリスト（スキーマオブジェクトもしくはインスタンス）の記憶アドレスがステップ 1003 で確かめられる。一時的な仮想メモリアドレスは、与えられたハンドルをもつオブジェクトが削除さ

れることを指示するために保持される。削除されるオブジェクトのみがこの特別メモリアドレスを持つことを許容される。もし、ステップ1003におけるハンドル参照（ルックアップ）で見つけられるアドレスが削除されたオブジェクトアドレスであれば、その時エラー条件が生成され、そしてハンドルマネージャ137はステップ1004でNULL仮想メモリアドレスをダイナミッククラスマネージャ234に戻す。

そうでなければ、ハンドルマネージャ137がステップ1005で続く。もし、ステップ1005のリストにおいて見つけられた仮想メモリアドレスが、NULLポインタでなければ、その時、処理がステップ1009で続く。もし、ステップ1005で見つけられた仮想メモリアドレスがNULLであれば、その時、要求されたオブジェクトはメモリに表れない。ハンドルマネージャ137は、与えられたハンドルを持つオブジェクトを第2記憶103から読むことをファイルマネージャ140に要求し、仮想アドレス空間にオブジェクトを生成し、そしてステップ1006で仮想メモリアドレスをハンドルマネージャ137に戻す。

ステップ1007において、ファイルマネージャ140により作られたオブジェクトの仮想メモリアドレスが特別な削除仮想メモリアドレスに対してテストされる。もし、ファイルマネージャ140がオブジェクトが削除されていることを判定したなら、その時エラー条件が生成され、そしてNULLポインタがステップ1008で戻される。そうでなければ、ステップ1009で処理が続く。

ステップ1009で、ハンドルマネージャ137が、与えられたハンドルをもつオブジェクトに対する有効仮想メモリアドレスを識別する。オブジェクトのタイプは、それがハンドルのタイプと同じタイプであることを確認するためにテストされる。もし、タイプが正しければ、その時エラー条件が生成され、そしてNULLポインタがステップ1010で戻される。そうでなければ、要求されたオブジェクトのアドレスが識別され、そしてそのアドレスはステップ1011でダイナミッククラスマネージャ134に戻される。

ダイナミッククラスマネージャ134が、API143の機能の半分に新しいスキーマオブジェクトもしくはインスタンスを作る時、ハンドルマネージャ137は、新しいオブジェクトに対する新たなハンドルを生成するために援用される。

ハンドルマネージャ 137 は、予め生成された最大ハンドルにシークエンシャルに続く次のリストのインデックスである使用されていないハンドルを戻す。言い換えれば、それは、オール最大値+1に戻す。ハンドルマネージャ 137 は、それがリストに入るように新しいオブジェクトのアドレスについて情報を受け取る。

ハンドルマネージャ 137 は、オブジェクトが API 143 の機能の半分で削除される時にはいつでも、ダイナミッククラスマネージャ 134 により援用される。与えられたハンドルによりインデックス付けされるリストの仮想メモリアドレスは、特別な削除オブジェクトアドレスをセットされる。

5. ユニットマネージャ

ユニットマネージャ 138 は、ニューメリックアトリビュートの生成、メンテナン스와およびベースおよび派生されたユニットの削除に対するサービスを提供するダイナミッククラスマネージャ 134 の欠くことのできない部品である。その与えられたデータストラクチャーは、ダイナミッククラスマネージャ 154 の記述の重要な部分として詳細に議論される。ユニットマネージャ 138 の存在、および数量が測定されるユニットに数量を関係付け、そしてデータベースにおいて数値を更新し、サーチそしてソートする時に互換性のあるユニット間で自動的に変換することを実行するその能力の存在は、ユニットに欠けている与えられる数値を記憶することに比較して重要な利点である。

6. ファイルマネージャ

ファイルマネージャ 140 は、スキーマオブジェクトとインスタンスに対する第2記憶機構 103 へのアクセスを与える知識ベースサーバ 132 のサブシステムである。ファイルマネージャ 140 は、知識ベース 123 を読むこと、書き込むこと、更新すること、生成することおよび参照するためのサービスについての独立セットであるアクセスソメソッドを与える。

ファイルマネージャ 140 は、知識ベースオブジェクトの永久記憶 103 へのアブストラクトインタフェースをダイナミッククラスマネージャ 134 およびハンドルマネージャ 137 に与える。言い換えれば、ファイルマネージャ 140 は、派生クラスにより十分に定義されるように意図されている C++ アブストラク

トベークラスである。ファイルマネージャ 140 により提供されるインタフェース機能もしくはメソッドは、テーブル 5 とテーブル 6 に示される。ファイルマネージャ 140 により与えられる機能は、その利用に依存するグループに分離できる。

第 2 記憶を開きそして閉じる機能は、クラスマネージャ 134 が知識データベース 123 のサービスを生成する時、知識ベースサーバ 132 が開始する時もしくは知識ベースサーバ 132 が終了する時に、クラスマネージャ 134 により使用される。クラスマネージャ 134 は望ましい配置で知識ベースサーバ 134 を初期化するためにウォームスタート機能を使用する。ファクトリ生成機能がファイルマネージャファクトリにより使用される。技術に熟達した者は、オブジェクトの具体化に対するファクトリの使用について良く知っていて、そのような機能は、詳細には説明しない。コプリアン (Coplien) の ”Advanced C++” を参照のこと。

他のファイルマネージャ 140 は、知識ベース 123 を修正するある機能を実行する時にはいつも、ダイナミッククラスマネージャ 134 により使用される。これらの機能は API 134 および知識ベース 123 を修正するダイナミッククラスマネージャ 134 機能に対してほとんど一対一に対応する。ファイルマネージャ 140 は、第 2 記憶 103 のデータがダイナミッククラスマネージャ 134 のデータを正確にモデルすることを保証することに責任がある。

付加的なファイルマネージャ機能は、ダイナミッククラスマネージャ 134 が仮想メモリにない任意のオブジェクトのハンドルを使用する時に、ハンドルマネージャ 137 により使用される (第 5 図, ステップ 1006 参照)。これらの機能は、第 1 記憶 103 からオブジェクトを読むことにより仮想記憶にオブジェクトを作る。生成されたオブジェクトのアドレスはハンドルマネージャ 137 に戻される。

表 5

A P I 機能に対してクラスマネージャにより使用される 機能

```
virtual long getDBFeatureCode
```

```

virtualconst cd- stringList CD- FAR & getDBCcopyright
virtual cd - boolean addLeafClass
virtual cd - boolean addInstance
virtual cd - boolean addAttribute
virtual cd - boolean addEnumerator
virtual cd - boolean addStringArrayElement
virtual cd - boolean changeSubclassOrder
virtual cd - boolean changeAttributeOrder
virtual cd - boolean changeEnumeratorOrder
virtual cd - boolean deleteLeafClass
virtual cd - boolean deleteInstances
virtual cd - boolean deleteAttribute
virtual cd - boolean deleteEnumerator
virtual cd - boolean deleteStringArrayElement
virtual cd - boolean deleteMoveInstance1
virtual cd - boolean moveAttribute
virtual cd - boolean moveSubtree
virtual cd - boolean setParameter
virtual cd - boolean setStringArrayElement
virtual cd - boolean setParameterUndefined
virtual cd - boolean setClassName
virtual cd - boolean setAttributeName
virtual cd - boolean setEnumeratorName
virtual cd - boolean setClassCoder
virtual cd - boolean setAttributeRequired
virtual cd - boolean setAttributeProtected
virtual cd - boolean addUnitFamily
virtual cd - boolean addUnit
virtual cd - boolean setEnumeratedUnitRows

```

```

virtual cd - boolean setClassMetaParameters
virtual cd - boolean setAttributeMetaParameters
virtual cd - boolean setClassEnumeratorMetaParameters
virtual cd - boolean setUnitMetaParameters
virtual cd - boolean setEnumeratedUnitTable
virtual cd - boolean setUnitFamilyName
virtual cd - boolean setUnitName
virtual cd - boolean setUnitConversionValues
virtual cd - boolean deleteDerivedUnit
virtual cd - boolean setAttributeUnits

```

第2記憶を開くおよび閉じるための機能

```

cd - fileManager
virtual -cd - fileManager

```

ウォームスタートのための機能

```

virtual -cd - class CD - FAH * warmStart

```

派生したファイルマネージャのファクトリ生成のための機能

```

static cd - fileManager1 * make

```

メモリにオブジェクトを記憶するハンドルマネージャにより使用される機能

```

virtual void getClass
virtual void getAttribute
virtual void getEnumerator

```



```
virtual void getUnit
virtual void getUnitFamily
virtual void getShemaObject
virtual void getInstance
```

現在において望ましい実施例は、ベースファイルマネージャクラスから派生したファイルマネージャ140を含む。ヌルファイルマネージャ140（“nullmgr.hxx”と称される）はファイルマネージャ140の機能の全てを定義するが、これらの機能のいくつかの効果はヌル（null）である。ヌルファイルマネージャ140は、ダイナミッククラスマネージャ134に対して第2記憶を提供する。ファイルマネージャ140のこのタイプの目的は、第1にテストすることである。

ファイルマネージャ140の第2の派生は、カディスファイルマネージャ（Cadis File Manager）である。カディスファイルマネージャはスキーマオブジェクトとインスタンスオブジェクトの永久記憶に対する第2記憶と相互作用する。第2記憶に蓄積されるようなファイルのフォーマットは、第53図-第64図に示される。カディスファイルマネージャは、また、マップされた1/O、標準1/O、および行1/Oアクセス方法を管理する。

カディスファイルマネージャは、第2記憶103の単純ファイルにおける知識ベース123のカレントステートのコピーを保持する。第2記憶のコピーは単一知識ベース123として考えられるが、便宜的に、それは第2記憶上の3つのファイルにマップされる。これらの3つのファイルは、スキーマファイル、インスタンスファイルおよびダイナミックファイル2400として知られる。スキーマファイルおよびインスタンスファイルは、スキーマオブジェクトとインスタンスオブジェクトのそれぞれについての固定的サイズのデータを含む。ダイナミックファイルは、性質により長さが変化するアイテムおよびキャラクタストリングのリストのようなデータを含む。

第53図を参照すると、ダイナミックファイル2400のシークエンシャルファイル2400が示されている。ダイナミックファイル2400は、ヘッダ2401（以下に説明される）および、シークエンシャルに続いて、複数の可変長オ

プロジェクトを含む。第1の可変長オブジェクトは2402、第2の2403である。第1の可変長オブジェクトは、2402、第2の2403である。これらのオブジェクトは、ファイルの中で下に続く。第54図は、スキーマとインスタンスファイル2404の一般的レイアウトを示す。ここに、フォーマットは、スキーマ2401とオブジェクト2405、2406のシリーズ等と本質的に同じである。スキーマおよびインスタンスファイルの各々において、しかし、オブジェクトは、公知のサイズである。これは、オブジェクトの中のその順序を示す位置を知るだけで、ファイルに高速に位置付けることができることを意味する。現在の実施例において、この順序を示す位置は、オブジェクトに指定されたハンドルの値に正確に必ず等しい。

第55図は、3つのファイル全部にあるファイルヘッダ2401のレイアウトを示す。3つのファイルのヘッダの最初の6つのコンピュータ記憶ワードはファイルを横切る方向に同じフォーマットで続く。これらの6つのワードは、知識ベース123の開放数2407と改訂番号2408、知識ベースが生成された時のデータ2409、データを含むファイルの最後の位置のファイルオフセット2410、知識データベース123が更新されるかどうかを指示するブーリンフラグ (`boolean flag`) 2411、ファイルにあるデータのソースを選択的に指示する特徴コード2412、および2つのフィラーワード2413と2414を含む。3つのファイル全部のタイプのヘッダは、その時、2つの付加的なワードを含む。それらのワードの内容は、ファイルの中で様々であろう。スキーマファイル2404は、グローバルユニットのリストがあるダイナミックファイル2400へのオフセット2415およびスキーマファイルで使用するハンドルの最大値2416を含む。インスタンスファイルは、付加的なフィラーワード2417とインスタンスファイル2418で使用するハンドルの最大値を含む。ダイナミックファイルは付加的なフィラーワード2419と値“1”を含むワード2420を含む。

スキーマファイルにあるオブジェクト2405、2406等は、スキーマファイルにある様々なタイプに対応する様々なタイプのオブジェクトである。大部分の場合、それぞれのタイプのオブジェクトに対する知識ベースに記憶される値は

、ダイナミッククラスマネージャ 1 3 4 によりメモリに保持される値と非常に直接的に対応する。第 5 6 図、第 5 7 図、第 5 8 図、第 5 9 図および第 6 0 図は、これらの様々なオブジェクトのレイアウトを示す。これらのオブジェクトのタイプの各々は、1 2 のコンピュータ記憶ワードから構成される。

第 5 6 図は、知識ベース 1 2 3 のクラスにあるスキーマファイルオブジェクト 2 4 2 1 のレイアウトを示す。クラスオブジェクトは、クラスが削除されているかどうかを示すフラグ 2 4 2 6、常に“2 0”であるタイプコード 2 4 2 7、クラスが“プライマリ”、“セコンダリ”もしくは“コレクション”クラスであるかどうかの指示 2 4 2 8、エンブディフィアードバイト 2 4 2 9、クラスのハンドル 2 4 3 0、クラスのハンドル 2 4 3 0、その親クラスのハンドル 2 4 3 1、クラスに属するサブクラスのリストが見出されるダイナミックファイルへのオフセット 2 4 3 4、クラスのローカルアトリビュートのリストが見つけれられるダイナミックファイルへのオフセット 2 4 3 3、クラスに属するインスタンスのリストが見出されるダイナミックファイルへのオフセット 2 4 3 4、クラスにルートされるサブツリーに現在位置されるインスタンスの番号 2 4 3 5、クラスに属するメタパラメータのリストが見つけれられるダイナミックファイルへのオフセット 2 4 3 6、3 つのフィラードワード 2 4 3 7、2 4 3 8 と 2 4 3 9 およびクラスの名称が見出されるダイナミックファイルへのオフセット 2 4 4 0 を含む。

第 5 7 図は、知識ベースにおけるアトリビュートを表すスキーマファイルオブジェクト 2 4 2 2 のレイアウトを示す。アトリビュートオブジェクトは、オブジェクトが削除されるどうかを示すフラグ 2 4 4 1、計算アトリビュートに対して 5 1、ブーリアンアトリビュートに対して 5 1、ニューメリックアトリビュートに対して 5 2、ストリングアトリビュートに対して 5 4 およびストリングアレイに対して 5 5 であるタイプコード 2 4 4 2 を含む。それは、アトリビュートが“要求された”かどうかを指示するフィールド 2 4 4 3、それが“プロテクテッド”かどうかを示すフィールド 2 4 4 4、アトリビュートのハンドル 2 4 4 5、およびアトリビュートを定義するクラスのハンドル 2 4 4 6 を含む。もしアトリビュートが計算アトリビュートであれば、アトリビュートに属する演算子ハンドルのリストが見出されるダイナミックファイルへのオフセット 2 4 4 8 が存在する。

もし、アトリビュートがニューメリックアトリビュートであれば、ニューメリックアトリビュートが使用するユニットを含むユニットファミリーに対するハンドル2449が存在するであろう。もし、アトリビュートがそれらの2つのタイプの一つでなければ、フィラーワード2447が存在するであろう。アトリビュートは、このアトリビュートに対するメタパラメータがリストされるダイナミックファイルへのオフセット2450、およびフィラーワード2451、2452、2453、2454および2456を含む。最後に、アトリビュートはアトリビュートの名称が与えられるダイナミックファイルへのオフセット2457を含む。

第58図は、知識ベース123の演算子を示すスキーマオブジェクト2423のレイアウトを示す。演算子オブジェクトは、オブジェクトが削除されたかどうかを示すフラグ2358、常に“60”を含むタイプコード2459、2つのフィラーバイト2460、演算子のハンドル2461、演算子に対するメタパラメータが見出されるダイナミックファイルへのオフセット2462、2463から2470までのフィラーワードおよび演算子の名称が位置されるダイナミックファイルへのオフセット2471を含む。

第59図は、知識ベースにおけるユニットを表すスキーマファイルオブジェクトのレイアウト2424を示す。ユニットオブジェクトは、オブジェクトが削除されたかどうかを示すフラグ2472、ベースユニットに対して“81”、実際にドライブされるユニットに対して“91”および数値化テーブルに対して“92”であるタイプコード2473、ユニットが整数あるいは実数あるいは計算されたかどうかを示すユニットタイプフラグ、計算ユニットに対して、テーブルに示されるべき行数を含むフィールド2475、ユニットのハンドル2476、このユニットを定義するユニットファミリーのハンドル2477およびこのユニットが派生するベースユニットのハンドル2478（即ち、もしこれがNULLであれば、ユニットはベースユニットである）を含む。ベースユニットに対して、2つのフィラーワード2479と2480が存在するであろう。実際に派生されたユニットは、複数のファクタ2481とオフセット2482をもつ。数値化テーブルは演算子ストリングのリストが位置されるダイナミックフィールドへの

オフセット 2 4 8 3 および実数値のリストが位置されるダイナミックフィールドへのオフセット 2 4 8 4 をもつ。あらゆるユニットタイプは、それから、メタバレータリストが見つけられるダイナミックフィールドへのオフセット 2 4 8 5、4 つのフィルターワード 2 4 8 6 - 2 4 8 9 およびユニット名に対するダイナミックファイルへのオフセット 2 4 9 0 を持つ。

第 6 0 図は、知識ベースにおけるユニットファミリを表すキーマファイルオブジェクトのレイアウト 2 4 2 5 を示す。ユニットファミリオブジェクトは、オブジェクトが削除されたかどうかを示すフラグ 2 4 9 1、常に“7 0”であるタイプコード 2 4 9 2、2 バイトフィルターフィールド 2 4 9 3、ユニットファミリのハンドル 2 4 9 4、このユニットに含まれるユニットファミリハンドルのリストが見つけられるダイナミックファイルへのオフセット 2 4 9 5、およびこのファミリに定義されるユニットハンドルのリストが見つけられるダイナミックファイルへのオフセット 2 4 9 6、7 つのフィルターワード 2 4 9 7 - 2 5 0 3、およびユニットファミリの名称が見つけられるダイナミックファイルへのオフセット 2 5 0 4 を含む。

インスタンスファイルにあるオブジェクト 2 4 0 5、2 4 0 6 等は、全てインスタンスオブジェクトである。各インスタンスオブジェクトは 4 つのコンピュータ記憶ワードから構成される。第 6 1 図はインスタンスファイルオブジェクト 2 5 1 1 のレイアウトを示す。インスタンスオブジェクトは、インスタンスが削除されたかどうかを示すフラグ 2 5 0 2、常に“3 0”であるタイプコード、2 バイトフィルターフィールド 2 5 0 7、インスタンスのハンドル 2 5 0 8、インスタンスに依存するクラスのハンドル 2 5 0 9、およびインスタンスに属するパラメータのリストが見つけられるダイナミックファイルのオフセット 2 5 1 0 を含む。

ダイナミックファイルにあるオブジェクト 2 4 0 2、2 4 0 3 等は、そこに記憶されるコンポーネントサイズに基づく様々なタイプをもつ可変長である。第 6 2 図は、キャラクタストリングを記憶するのに使用されるタイプ 1 のダイナミックオブジェクト 2 5 1 2 のレイアウトを示す。タイプ 1 のダイナミックオブジェクトは、それが削除されたかどうかを示すフラグ 2 5 1 6、“1”であるタイプコード 2 5 1 7、記憶キャラクタストリングの長さ 2 5 1 8、キャラクタストリン

グに対するファイルに位置付けされる実際のスペース量2519, 2バイトフィルター2520, および記憶ストリングを含むキャラクターのブロック2513を含む。第63図は、ハンドル、整数、実数、オフセットのような4バイト長の記憶データアイテムに使用されるタイプ2のダイナミックオブジェクト2514のレイアウトを示す。タイプ2のダイナミックオブジェクトは、それが削除されたかどうかを示すフラグ2521, “2”であるタイプコード2522, 2バイトフィルター2523, 記憶データの長さ2524, ファイルにおいてデータを実際に配置する空間量, および実際の記憶値を含むデータブロック251を含む。第64図は、パラメータデータを記憶するのに使用されるタイプ3のダイナミックオブジェクトのレイアウト2526を示す。各記憶パラメータは、4コンピュータワードである。タイプ3のダイナミックオブジェクトは、オブジェクトが削除されたかどうかを示されるフラグ2527, “3”であるタイプコード2528, 記憶データ長2529, データを実際に配置する空間量2530, 2バイトフィルター2531, および連続するパラメータオブジェクト2532, 2547等を含む。各パラメータオブジェクト2432は、パラメータが削除されたかどうかを示すフラグ2533, パラメータが計算されるものか、ブーリン (Boolean), ニューメリック, スtringもしくはStringアレーであるかどうかを示すタイプコード2534, 2バイトフィルター2534およびこのパラメータが参照するアトリビュートのハンドル2536を含む。

パラメータが計算タイプであるなら、パラメータオブジェクトはパラメータがセットされる演算子のハンドル2537およびフィルターワード2538を含む。もし、パラメータがブーリンタイプのものであるなら、パラメータオブジェクトは、実際に記憶されたブーリン値2539およびフィルターワード2540を含む。もし、パラメータがニューメリックタイプであるなら、パラメータオブジェクトは、値が表現されているユニットのハンドル2541およびそれらのユニットに表現されている実際の数値2542を含む。もし、パラメータがStringタイプのものであるなら、パラメータはString値が位置されるダイナミックファイルへのオブジェクト2543およびフィルターワード2544を含む。もし、パラメータがStringアレータイプのものであるなら、パラメータは、記憶さ

れたキャラクタストリングへのオフセットのリストが見つかるダイナミックファイルへのオフセット 2 5 4 5 およびファイラーワード 2 5 4 6 を含む。

7. データベースマネージャ

データベースマネージャ 1 3 9 は、知識ベースサーバに 1 3 2 により管理される知識ベースについてのハイレベル情報を記憶しそして管理する知識ベースサーバ 1 3 2 のサブシステムである。データベースマネージャ 1 3 9 により保持されるデータのグラフィカル表現が第 1 5 2 図に示されている。データベースマネージャ 1 3 9 は、知識ベースサーバ 1 3 2 により管理される知識ベース 1 2 3 についてのエントリーのリンクされたリストを保持する。

データベースマネージャ 1 3 9 は、データベースオブジェクトの競合コントロールに対して責任がある。競合コントロールのために、書き込みロックがクラスに保持される。書き込みロックは、読み出しもしくは検索操作がロックされたクラスに実行するが、更新操作はロックホルダーによって実行されるだけである。ロックは、レガシーワークエリアの競合更新者とブライバシーを許容するためにスキーマエディター 5 0 0 およびレガシー（遺産）1 3 3 によってセットされる。

。クラスあたりに一個だけロックホルダーが認められる。ロックホルダーは、ユーザ名ではなく、その接続により識別される。ロックは、接続長を保持される。一度、接続が知識ベース 1 2 3 を閉じるかもしくは接続タイムアウトのためのいずれかにより接続が壊されると、そのコネクションによりホールドされる全ロックが開放される。

クラスをロックすることは、そのクラスにより定義される全アトリビュートにロックすることである。ロックはインスタンスを編集するための補助である。

ロックすることのグラニュラリティ（粒状化）は、知識ベース、ツリーおよびクラスレベルにおいてである。ロックはクラスにローカルにセットされ、あるいはインヘリットされる。ローカルクラスのロックは、クラスロック機構を使用してセットされる。これらは、ロックされたクラスのサブクラスによりインヘリットされないローカルロックである。例えば、知識ベース 1 2 3 のルートクラスは、更新を妨げるためにロックされるクラスであり、サブクラスは、しかし、他の

ユーザによりロックされる。

ロックは、知識ベース123において全クラスを是非を問わずロックする知識ベース123をロックすることによりインヘリットされる。ロックはサブツリーロックすることによりインヘリットされる。サブツリーはツリーロックをクラスに適用することによりロックされる。ツリーロックされたクラスの全降順クラスは、暗黙のうちにロックされる。物理的に、サブツリーの任意のクラスロックは、サブツリーもしくは知識ベースロックにより包含される。ツリーロックを得ようとするユーザに対して、そのツリーのノードは、他のユーザによりロックできない。

ロックオブジェクトのグラニュラリティのものと詳細な議論は、ACM、フレズにより発行されたワン キム (W o n K i m) , “オブジェクト指向データベース”,あるいはワン キム, “オブジェクト指向概念, データベースおよびアプリケーション”, (1989)を参照すること。

第86図において、クラスBがユーザ1によりロックされるとしよう。ユーザ1は、他のユーザによりツリーにおいてホールドされるロックが存在しないので、クラスAのツリーロックを与えられる。他の例において、クラスBがユーザ1によりロックされるとしよう。ロックは、クラスC、クラスD、クラスEおよびクラスFに対してユーザ2に与えられる。なぜなら、それらのクラスに対しては、別のロックホルダーが存在しないからである。ユーザ2は、クラスAのローカルツリーロックに与えられるが、ツリーロックは、ユーザ2を拒否される。というのは、クラスBはユーザ1によりロックされているからである。

この発明の一特徴は、インタフェースは、インタフェースへのパラメータであるアトリビュートもしくはクラスのロックを必要とすることを具体化できるということである。データベースマネージャ139は、少なくともデータベースマネージャレベルにおいてロックをチェックし、そしてそれは、後に続いて起こるロック衝突を解決することでクラスマネージャ134を助ける。

D. API

アプリケーションプログラミングインタフェース即ちAPI143は、知識サーバ132、レジストリサーバ141により提供される機能にアクセスする外部

的なCもしくはC++言語機能、およびクライアントアプリケーション130、133および144へのライセンスマネージャ142機能を参照する。

E. レジストリサーバ

レジストリサーバ141は、ユーザと知識ベースに対する管理および安全機能を提供するUNIXプロセスである。ユーザ管理機能は、名称とパスワード管理と知識ベース123へのユーザアクセス権のマッピングを含む。レジストリサーバにより提供される知識ベース管理はRPCサーバビスマッピング、ホストCPUマッピング、および物理ネームマッピングへの論理を含む。

F. ライセンスマネージャ

ライセンスマネージャ142は、ソフトウェアおよびライセンスされた知識ベース123に対するソフトウェアライセンス制御を与えるUNIXサーバプロセス（図示の例において、“pma1m”と称されている）である。ライセンスマネージャ142の満足のゆく操作は、エランコンピュータグループ株式会社から入手できる慣用的なエランライセンスマネージャを使用して達成できる。

G. スキーマエディタ

スキーマエディタ144は、スキーマオブジェクトを生成、編集および削除するためのグラフィカルインタフェースを提供するアプリケーションである。オブジェクトは、リネーム、リオーダおよび移動される。スキーマエディタ144は、API143を使用する知識ベースクライアント131と通信する。スキーマエディタ144は、オブジェクト指向グラフィカルユーザインタフェースを提供する。ユーザはキーボード115とマウス114を介する入力を与えるスキーマエディタ144と相互に関係する。

第87図は、ユーザがシステムにログオンに成功した後にディスプレイ116のスクリーンに表れ、そして第88図に示されるパーツ専用ウィンドウによるブルダウンメニュー146からスキーマエディタを選択する通常のディスプレイを示す。ここで説明される特定の例は、ウィンドーズ環境で説明され、それは、本発明がウィンドーズの実施例に限定されないことが理解されるものとしてである。技術に熟達したものは、マウス114によりクリック、ダブルクリック、ドラッグ、ポインタおよび選択をいかにするかを含むウィンドーズ技術および指示を

良く知っている。付加的な情報は、マイクロソフト通り、リドモンド、ワシントン、98052-6399、マイクロソフト株式会社のパーツ番号21669から入手できるマイクロソフトウィンドーズユーザズガイドから入手できる。

ユーザが最初、スキーマエディタ144を開くと、第89図に示されるようにスキーマ編集ウィンドウ500が現れる。最初、スクリーン501の左手位置は、選択されたクラスのタイトルを変更するために使用されるクラスタイトル編集ボックス502を表示する。クラスタイトルOKボタン503とチャネルボタン504はクラスタイトル変更を受け取るか、もしくはリジェクトするのに使用される。クラス付加ボタン505および削除ボタン506はクラスを付加および削除するために使用される。スクリーン501の左手位置に、また、ルートクラス507およびルートサブクラス508が表示される。図示の例において、ルートサブクラス508は“電子部品”、“機械的”および“材料”である。ルートクラス507は、親を持たない最上位クラスである。この例において、知識ベース123の名称もしくはスキーマの最初である。サブクラス508は、親をもつクラスである。クラス507が選択された時、そのクラス507に属するなんらかの任意のサブクラス508がディスプレイ501に現れる。サブクラスは親の子である。例えば、機械的サブクラス508はルートクラス507であり、機械的サブクラス508は親ルートクラス507の子である。第89図に示される例において、3つのサブクラス508が存在する。

スクリーン509の右手位置は、ルートアトリビュート516を表示する。図示の例において、アトリビュートは、“パート番号”、“説明”および“費用”である。アトリビュート516は、クラスもしくはサブクラスのキャラクタ507である。アトリビュート番号列517は、スクリーン501のクラスサイド上に示される選択されたクラスに対するローカルおよびインヘリットされた双方の全アトリビュートを表示するのに使用される。ロックコラム519および要求された列520は、ロック（プロテクトされて）されたもしくはは要求されたアトリビュートに使用される。ユーザは、ロックコラム519の望むアトリビュートの行あるいは必要な列520をクリックし、もしロックもしくはは要求したものがターンオンされれば、チェックマークが選択した行／列に現れる。ロックあるいはは

要求されたアトリビュートはリトソーバ130の記述に關係して上記に説明した部品を作るために使用される。また、選択されたアトリビュートのタイトルを変更するために使用されるアトリビュートタイトル編集ボックス510がスクリーン509の右手位置に表示される。アトリビュートタイトルOKボタン511とキャンセルボタン512がアトリビュートタイトル変更を容認あるいは拒絶するのに使用される。アトリビュート付加ボタン513、削除ボタン514および編集ボタン515はあるアトリビュート付加、削除もしくは編集するのに使用される。これらのボタンのコマンド名は、ユーザがエリア501の選択クラスにより所有されていないアトリビュートを選択した時に、暗くなる。編集ボタンは、また、もしアトリビュートタイプが数値でないもしくは計算されていないのいずれかであれば暗くされる。

クラスツリー508は、第90図のフローチャートおよび第91図に関連して説明されるように、閉じたフォルダのアイコン189をダブルクリックすることによりナビゲートされる。ステップ521でユーザが閉じたフォルダのアイコン529上をダブルクリックすると、オープンフォルダが表示され、そしてサーチクラスのリストがステップ522で得られる。得られた各サブクラスに対して、アイコン531、532がステップ524のリーフクラス531もしくはステップ525のサブクラス532を表すように表示される。アトリビュートはエリア509の選択クラスに対して表示され、制御がステップ528でユーザに戻される。クラスがフォルダアイコン190のダブルクリックにより閉じられ、これは、閉じたフォルダアイコン529を表示し、そして選択されたクラスの全サブクラスを合体する。リーフクラスは、なんらのサブクラスを持たず、そしてドキュメントアイコン531として表示される。リーフクラス531はオープンもしくはクローズできない。

クラスは、第92図のフローチャートおよび第93-94図に關係して説明されるように、新しいサブクラスに再度表となることができる。ユーザはサブツリーを選択し、ステップ534で移動され、スクリーンエリア501がハイライトされる。ステップ535において、ユーザは、マウスボタン117の押し下げおよびキーボード122のコントロールキーをホールドし、そしてドラッグされて

選択されたクラスの新しい親になるべきクラス544上の領域501にドラッグする。ユーザは、選択されたクラスをドラッグする時、ドラッグされたクラスはハイライトされ、そして、もし、クラスがステップ535でドラッグされる選択クラスの兄弟姉妹であるなら、マウスカーソルはステップ538でドロップされないアイコンに変更される。もし、ドラッグされて選択されたクラスがドラッグされて選択されるクラス544の兄弟姉妹でないなら、ステップ539でカーソルはドロップアイコンに変更される。ユーザが、ステップ539において正しいドロップアイコン上をドラッグされた選択クラスにドロップした時、知識ベース123は、ステップ541で新しいクラス構成を示すように更新される。また、クラスツリー501は、クラスツリー542および545を表すように更新される。制御はそれからユーザ528に戻される。

クラスは、第95図のフローチャートおよび第96図-第97図に関係して説明されるように、兄弟姉妹をもつサブクラスの中にアレンジできる。ユーザは、スクリーンエリア501でステップ547において再アレンジされるサブツリー545を選択する。ユーザはマウスボタン117を押し下げて保持し、そしてステップ547においてドラッグされて選択されるクラスの新しい位置になるようにクラス上の領域501にクラスをドラッグする。ユーザが選択されたクラスをドラッグする時、ドラッグされるクラスはハイライトにされ、そして、もしクラスがステップ547においてドラッグされる選択クラスの兄弟姉妹でなければ、ステップ551においてドロップされないアイコンに変更される。もし、ドラッグされるクラスがステップ547においてドラッグされる選択されたクラスの兄弟姉妹なら、カーソルはステップ552においてドロップアイコンに変更される。ユーザが、正しいドロップクラス上にステップ553においてドラッグされて選択されたクラスをドロップする時、知識ベース123は更新されて、ステップ554において新しいクラスストラクチャーを表示する。クラスツリー501が、また更新されて、ステップ555で新しいクラスツリーを表示する。制御は、それからステップ528でユーザに戻される。

新しいクラスは、第98図のフローチャートと第99-100図との関係において説明されるように付加ボタン505を使用して付加される。ユーザは、付加

されるべきクラスの親として使用されるクラスツリー領域501のクラスを選択する。ユーザは付加ボタン505を選択し、そしてステップ560において、ダイアログ564が現れる。新しいクラスタイトルがステップ560において付加ダイアログボックスに入力される。この例において、“カスタムハードウェア”は、テキストエントリフィールド565に入力される。それから、ユーザは、OKボタン566もしくはチャネルボタン567のいずれかを選択する。もし、ステップ561において、OKボタン566が選択されると、新しいクラスが知識ベースに付加される。スクリーン501は、第100図に示されるようにステップ562で新しいクラスツリー568に付加される。新しいクラスは、リーフクラスであり、そしてドキュメントアイコン531のように表示される。もし、親クラスがリーフクラスであれば、親クラスアイコンはオープンフォルダアイコン530に変更される。付加クラスダイアログ564がステップ563で閉じられ、そして制御がステップ528でユーザに戻される。もし、キャンセルボタンが選択されたら、付加クラスダイアログボックス564がステップ563で閉じられ、そして制御がステップ528でユーザに戻される。

アトリビュートは、第101図のプローチャートにおいておよび第102図と第103図に関係して説明される。この例において、“フィニッシュ”579は、“ヘッドリセッス”に再アレンジされる。ユーザは、スクリーンエリア509でステップ570で再アレンジされるべきアトリビュート579を選択する。ユーザは、マウスボタン117を押下げて保持し、そしてステップ576においてドラッグされて選択されるアトリビュートの新しい位置となるべきアトリビュート580上にアトリビュートエリア509のアトリビュート579をドラッグする。ユーザは、ステップ570で選択されたアトリビュートをドラッグする時、ドラッグされたアトリビュートがステップ570でハイライトされ、そして、もしクラスが継承されたアトリビュートであるなら、マウスカーソルはステップ574でドロップされないアイコンに変更される。ステップ573を参照。もし、ドラッグされるアトリビュートが継承さないアトリビュートなら、カーソルはドロップアイコン575に変更される。ユーザが正しいドロップアトリビュートをステップ576でドラッグされて選択されるアトリビュートをドロップする時、

知識ベース123は更新され、ステップ577で新しいアトリビュートストラクチャーを表示する。アトリビュートエリア509もまた更新され、第103図で示されるようにステップ578で新しいアトリビュートストラクチャー579を表示する。制御は、それから、ステップ528でユーザに戻される。

第105図に示されるように、第104図のプロチャートで説明されたものに新しい計算アトリビュートが付加される。この例において、“材料”としてタイトルされている新しい計算アトリビュートが付加されている。ユーザはスクリーンエリア509から付加ボタン513を選択する。付加アトリビュートダイアログ588がステップ582で表示される。ユーザは付加されるべきタイプのアトリビュートを選択し、この例において計算されたもの589がステップ583において選択される。ステップ584において、それからユーザは、計算アトリビュートを表示するようにアトリビュートタイトルを入力し、この例において、ユーザは“材料”590を入力した。ユーザはそれから、ステップ585においてOKボタンもしくはキャンセルボタンのいずれかを選択できる。もし、OKボタンが選択されれば、知識ベースは更新され、そしてエリア509のアトリビュートリストがステップ586で付加されたアトリビュートを含むように更新され、そしてステップ587で付加アトリビュートダイアログが閉じられる。ステップ528において制御がユーザに戻される。もし、キャンセルボタンが選択されれば、付加アトリビュートダイアログがステップ587で閉じられ、そして制御がステップ528でユーザに戻される。

第107図-第108図に示されるように、第106図のプロチャートに説明されたものにニューメリックアトリビュートが付加される。この例において、“レングス”とタイトルされたある新しいニューメリックアトリビュートが少量のユニットファミリーを使用して付加される。ユーザは、スクリーンエリア509の付加ボタン513を選択する。付加アトリビュートダイアログ588が、ステップ582において表示される。ユーザは、付加すべきアトリビュートのタイプを選択し、この例において、ニューメリック599がステップ594で選択される。ステップ584において、ユーザはそれから、ニューメリックアトリビュートを表すようにアトリビュートタイトル600を入力し、この例において、ユー

ザは“レングス”600を入力した。ユーザは、それから、ステップ585においてOKもしくはキャンセルを選択できる。もし、OKが選択されたら、ユニットファミリダイアログ1600がステップ595で表示される。ユニットファミリダイアログ1600は、ステップ595で表示される。ユニットファミリダイアログ1600は、完全な知識ベース123に対する利用可能な全ユニット1601のリストを含む。もし、OKボタン1602がこのダイアログボックス1600から選択されたら、ユニットタイプレングスの新しいニューメリックアトリビュートは知識ベースであり、そしてアトリビュートリストがステップ598で更新される。制御が、それからステップ528でユーザに戻される。もし、キャンセルボタン1603が選択されたら、付加アトリビュートダイアログ588がステップ587で閉じられ、そして制御がステップ528でユーザに戻される。

新しいブーリンアトリビュートが第109図のフローチャートに付加され、そして第110図で示される。この例において、“パーチェイスト”と題される新しいブーリンアトリビュートが付加されている。ユーザはスクリーンエリア509の付加ボタン513を選択する。付加アトリビュートダイアログ588は、ステップ582で表示される。ユーザは、付加されるべきアトリビュートのタイプを選択し、この例の場合、ブーリン1605と1607が選択される。ユーザは、それから、OKもしくはキャンセル585のいずれかを選択できる。もし、OKが選択されたら、知識ベースは更新され、そしてアトリビュートリストがアトリビュート586と509を含むように更新され、そして付加アトリビュートダイアログ587が閉じられる。キャンセルボタンが選択されたら、付加アトリビュートダイアログ587が閉じられ、そして制御がユーザ528に戻される。

新しいストリングアトリビュートは、第111図のフローチャートと第112図のスクリーンショットで説明される。この例において、マニフアクチャラーと題される新しいストリングアトリビュートが付加されている。ユーザは、スクリーンエリア509の付加ボタン513を選択する。付加アトリビュートダイアログ582と588は表示される。ユーザは、付加されるべきアトリビュートのタイプを選択するが、この例において、ストリング1609と1611が選択さ

れている。ユーザは、それから、ストリングアトリビュートを表示するようにアトリビュートタイトルを入力するが、この例の場合、ユーザはマニフアクチャラー-584と1610を入力した。ユーザは、それからOKもしくはキャンセル585を選択できる。もしOKが選択されれば、知識ベースは更新され、そしてアトリビュートリストが付加アトリビュート586と509を含むように更新され、そして付加アトリビュートダイアログ587が閉じられる。制御がそれからユーザ528に戻される。もし、キャンセルボタンが選択されれば、付加アトリビュートダイアログ587が閉じられ、そして制御がユーザ528に戻される。

第114図および第115図に示されるように、第113図のフローチャートに説明されているものに、計算タイプの演算子アトリビュートが付加および挿入される。計算アトリビュートは、スクリーンエリア509においてアクティブであり、編集ボタン515が活性化される。エディットボタン515が選択された時、エディット計算ダイアログボックス1620が選択された計算アトリビュートに対する演算子リストとともに表示される。ユーザは、付加ボタン1621もしくは挿入ボタン1622のいずれかを選択する。もし、付加ボタン1621がステップ1615で選択されたら、空白行が、ダイアログボックス1620の活性化演算子の後に付加され、そして知識ベース123が更新される。もし、挿入ボタン1622がステップ1616で選択され、空白行がダイアログボックスの活性演算子1620の前に付加され、そして知識ベース123が更新される。演算子タイトルがステップ1617でダイアログボックス1620の空白行にタイプ付けされ、この例の場合“アルミニウム”が入力され、そして知識ベースがステップ1617Aで更新される。第115図の例の場合、“スチール”が付加され、そして挿入ボタン1622が演算子“スチール”の上の空白行に付加するように選択される。ユーザが演算子の付加/挿入を終えた時、クローズボタン1623がステップ1619で選択され、そしてエディット演算子ダイアログボックス1620が閉じられる。制御がそれからステップ528でユーザに戻される。

計算タイプアトリビュートに対する演算子は、第117図に示されるように、第116部フローチャートにおける場合と同様に削除できる。計算アトリビュ

トがスクリーンエリア509でアクティブである時、エディットボタン515は活性化される。エディットボタン515が選択される時、エディット演算子ダイアログボックス1620が、選択された計算アトリビュート1613に対する演算子リスト1624とともに表示される。ユーザは、ステップ1626で演算子を選択し、それからステップ1627で削除ボタン1629を選択する。ステップ1627Aで、確認ダイアログボックス1630がステップ1627Cの“yes”ボタン1631もしくはステップ1627Dにおける“no”ボタン1632のいずれかのを選択することをユーザに可能にする。もし、“yes”が選択されたら、演算子はエディット演算子リスト1624から移動され、そして知識ベースがステップ1627Cで更新され、そして確認ダイアログがステップ1627Dで閉じられる。もし、ユーザがステップ1627Dにおいて“no”を選択したら、確認ダイアログが閉じられる。ユーザが演算子削除を終えたら、クローズボタン1623がステップ1619で選択され、そしてエディット演算子ダイアログ1620が閉じられる。制御がそれからステップ528でユーザに戻る。

第118図は、第119図においてニューメリックテーブルエディタダイアログボックスにより実行される機能を説明する。このダイアログボックス1550は、ニューメリックアトリビュートに対するニューメリック値のテーブルを作ることをユーザに可能にする。ニューメリックテーブルエディタダイアログ1550は、1552のようにニューメリックアトリビュートを選択した後スキーマエディター500からステップ1500で援用される。このエディットボタン515は、テーブルエディタダイアログ1551を援用する。

ステップ1501において、呼出しがAPI13を介して存在するテーブルデータを表示するためになされる。もし、テーブルデータが存在しないなら、1行1列のテーブルセル1554が作られそして第119図に示されるように表示される。

テーブル1554は数値とそれに関連するラベルをもつセルから構成される。ラベルは値とは別であり、そして隠された値の説明もしくは表示として使用される。テーブルセル1554は、昇順ニューメリック値を含まねばならない。ラベ

ルは照合順でよい。

第118図において、ユーザは、第120図に説明される手続きを実行することより、ステップ1504でのテーブルに値を付加する。ユーザは、任意に、第120図のステップ120においてテーブルに手作業でラベルすることができ、もしくは第121図のアイテム1559を選択することによりステップ1510のオートマティックラベル機能を援用できる。自動ラベルボタン1559はステップ1510でオートマティックバリュエダイアログボックス1560を行使する。ステップ1511において、ユーザは、アイテム1561、1562および1563に対する値を満たし、そしてOKボタン1564を選択する。ステップ1513において、セルに対する値が計算され、そしてテーブルにセットされる。ステップ1514において、オートマティックバリュエダイアログ1560が閉じられ、そして制御がステップ1515においてユーザに渡される。

第120図において、もし、ユーザがステップ1509においテーブルセルをマニュアルラベルすることを選択したら、ユーザは、アイテム1565を選択し、値を入力し、選択チェックボックス1566Aにより容認される。アイテム1566においてユーザが選択した任意のセルが、ステップ1517において値で満たされる。制御がステップ1515でユーザに戻される。

第118図において、ユーザはテーブルにラベルを付加するようにステップ1503を実行する。テーブルにラベルを付加するためのプロセスは、第122図に説明される。ステップ1519において、ユーザは自動ラベルもしくはマニュアルラベル付けを選択する。もし、第123図の自動ラベルアイテム1567が選択されたら、自動ラベル付けダイアログ1568がステップ1520において援用される。テーブル1569の各列に対して、ユーザはラベルのタイプを付けることができる。ステップ1522において、ユーザは、OKボタン1570もしくはキャンセルボタン1571を選択できる。もし、ユーザがOKボタン1570を選択すれば、ステップ1523は、自動ラベル付けダイアログからのラベル値で縦条に繋げられた現在セル値にセルラベルをセットする。ダイアログ1568が取り去られ、そして制御がステップ1524でユーザに戻される。

テーブルは行および列で構成される。ユーザはステップ1502を実行するこ

とによりテーブルの列と行数を変更しようとする。行と列はエディットボタンス1555もしくは1556およびチェックマークボタン1557を使用することにより入力される。値を受け入れられないように、“x”ボタン1558が選択される。列および行番号を変更するための手続きは、第124図に説明される。

第124図において、ユーザは、テーブルエディターダイアログ1550のアイテム1556を選択することによりステップ1536で行を選択する。ユーザはアイテム1556の行の番号を入力し、そしてアイテムを受け入れるためにアイテム1557を選択する。ステップ1537において、内部で受け入れられた行番号は、ステップ1536において入力される行番号に調整される。行番号はステップ1538でチェックされる。もし、行番号がテーブルにあらかじめきめられたある行番号より大きければ、新しい行がデフォルト値“0”でステップ540でラベルなしに付加され、ステップ1535でユーザに制御が戻される。もし新しい行が必要とされないなら、“0”もしくはより多い行がステップ1535において制御をユーザに戻す前に、ステップ1539においてそのラベルとともにテーブルから削除される。

第124図において、ユーザはアイテム1557を選択することによりステップ1530で変更する列を選択し、数値を入力し、そしてアイテムを受け入れるためにアイテム1557を選択する。この新しい番号は、最初にステップ1531でセットされる。もし新しい列がステップ1532で決定されるとして付加される必要があるなら、ステップ1534はテーブルにデフォルト値“0”およびラベルなしで付加する。もし、新しい列がステップ1532で決定される必要がなければ、“0”もしくはより多い列がそのラベルとともにテーブルから除去される。

第118図に戻って、ユーザは、第119図のOKボタンを選択することによりもしくは第119図のキャンセルボタンを選択することによって変更をキャンセルすることによりステップ1506のステップのテーブルエディタを閉じる。テーブルエディタダイアログボックス1550はステップ1507で消され、そして制御がステップ128でユーザに戻される。

第127図において、スキーマエディタ500でアトリビュートを削除するた

めの処理が示されている。第128図において、ユーザはメカニカルクラス2206を選択するエリア509における全セルは暗くなっている、削除ボタン519は活性化されない。スキーマエディタ500は編集されるべきメカニカルクラス2206により定義されるアトリビュートを受け入れるだけであり、そしてローカルに定義されたアトリビュートはない。

第129図において、ユーザはテストハードウェアアイテム2207を選択する。アトリビュートは、このクラスおよびエリア509において定義され、アイテム2208、2209および2210のローカルアトリビュートは暗くなくそして編集を可能にする。ユーザは第130図におけるアイテム2214を選択し、そしてハイライトされる。

第127図のステップ2200において、ユーザは第130図の削除ボタン519を選択する。

第127図のステップ2201において、第130図のダイアログボックス2211はアトリビュート2214の削除を検証することをユーザに可能にするように表示される。もし、ユーザが第130図のボタン2213を選択すれば、ダイアログボックス2211はステップ2204において消され、そして制御がステップ2205からステップ528にもどされる。

もし、ユーザが、第130図においてボタン2212を選択すれば、アトリビュートが、ステップ2203において知識ベースから削除され、そして第130図のアイテム2214が第130図のディスプレイエリア509から削除される。ステップ2204は、ダイアログ2211を消すことを実行し、制御がステップ2205からステップ528に戻される。

H. レガシおよびレガシマネージャ

レガシマネージャ145は、クラス化およびデータのパラメータ化をするためのダイナミッククラスマネージャ134の部品である。レガシ133は、部品をクラス化、パラメータ化、移動、移入および編集するためのグラフィカルインタフェースおよびツールを提供するアプリケーションであり、“レガシ処理”として知られる処理である。レガシ133はAPI143を使用する知識ベースクライアント131と通信する。

第132図は、カスタマレガシ部品データをダイナミッククラスマネージャ134により利用できる形態の知識ベース部品への変換を実行し、リトリーバ130を介してユーザにアクセスするための望ましいプロセスを示す。

ステップ600において、カスタマ部品データソース、それは、ファイル601の材料要求計画システム、部品マスタシステム、材料明細システム、購入システム、技術図面作成システム、部品カタログ、クリブ(crib)シート、インテリジェント部品ナンバリングシステムのデータを含むものであるが、レガシ133への入力として使われる入力ファイルに可能な限り完全もしくは部分的に含めるために解析される。レガシ処理において使用される部品データソースは、レガシ入力ファイル602に分離される。これらのオリジナル部品データソースは、固定長レコード、制限のないレコード、COBOLファイルフォーマット、もしくはレガシデータファイル604に移入可能にするためにステップ603で変換される他のもの、標準デリミタにより分離されるフィールドとの一致を識別部品により識別されたテキストから構成されるもの、通常ASCIIタイプキャラクタを含む。

ステップ605において、レガシ入力ファイルは、データ増加が適正であるかどうかを判定するために解析される。例えば、集積回路部品のクラスおよびパラータ情報がマニファクチャラーおよびデバイス番号により参照されて利用でき、ジェニック(生成)3000の使用によりそれらの部品の別の記述を増加もしくは置き換えられることに使用できる。どうように、クラス化およびパラメータ情報は政府工業規格もしくはカスタマ供給エンジニアリングテーブルを利用でき、それはこれらの部品の別の記述と自動的に結合される。結果の任意の増大した部品のレガシデータはファイル607に記憶される。

ステップ608は、任意に増大されたレガシデータ607の初期クラス化を実行するためのクラス化プログラムの実行を含む。加えるに、もし、部品データが部品データにおける識別パターンに基づいてクラスに移入されるべきであるなら、移入マップがその関連クラスのパターンの関係を記述して生成される。最終的に、ステップ608において、任意の必要なカスタムスキーマの開発がマニュアル手段の結合とスキーマプログラムの使用により実行される。ステップ608は

結果としてダイナミッククラスマネージャ 134 とレガシ 133 によりアクセス可能な予備的知識ベースを生成する。

ステップ 610 において、レガシ 133 のグラフィカルユーザインタフェースはクラス階層の識別サブツリーの申の部品の部品クラス化およびパラメータ化を実行することを与えられた主題事項の知識を備えユーザにより使用される。レガシ 133 の反復的応用により、予備的カスタム部品の知識データベース 611 が生成される。ステップ 612 において、ランダムサンプリング、部分的にクラス化された部品を識別するためのスキーマにおけるリフでないクラスの部品についてのクエリへのリトリバ 130 の能力の使用、不完全なパラメータ化を識別する未定義のアトリビュートについて質問（クエリ）すること、そしてソートすること、およびパラメータの最小、最大および標準値の検査の組合せが部品クラス化とパラメータ化の品質検査および修正に対して使用される。ステップ 612 において実行される品質保証活動の結果は、カスタムに配給するカスタム部品知識ベース最終的に用意される。ステップ 613 において、この知識ベースは、コンピュータテープ、ディスクもしくは他のコンピュータ読み出し可能手段、リトリバ 130 によりカスタムにより維持およびエンハンスされる知識ベース 614 とともに配給される。

本発明において、レガシ 133 は、部品機能 1101 をクラス化するためのグラフィカルユーザインタフェース、レガシマネージャ 145 の部品機能 1102 をパラメータ化すること、初期部品クラス化 3001 を実行するためのソフトウェアプログラムとともに、データからカスタムスキーマ生成のためのスキーマ生成プログラム 3002、そしてジェニク 3003、解析、ルックアップ、マニユファクチャラーと装置識別子に基づく集積回路タイプ部品に対するクラス化とパラメータ生成を提供する。

レガシ 133 は、クラス化およびパラメータ化される部品を選択することを含めて、質問すること、表示することおよび部品のパラメータを修正する手段として、およびクラス、アトリビュートおよびシソーラス編集のための演算子を選択するためにクラス階層および加算するアトリビュートをナビゲートする手段として、クエリ形成および部品表示およびリトリバ 130 の編集機能を含む。

レガシ133は、クラスに関連するメタデータとして記憶されるシソーラス入力の生成、修正および削除、ニューメリックアトリビュート、ブーリアンアトリビュート、計算アトリビュートの演算子、およびユニットファミリ内のユニットのためのグラフィカルユーザインタフェースを提供する。レガシ133は、また、クラスタイプの収集をセットし修正することのための手段、階層クラスを使用して部品機能をクラス化する制御に使用される第1および第2義的なものを含む。

それは、ソースアトリビュート1266、およびレガシマネージャ145により解析されるテキストパラメータを解析するテキストアトリビュートを選択する手段を含む。それは、また、方向アトリビュート1267を選択するための手段を備え、部品がクラス化されもしくはパラメータ化される時、テキストパラメータがシソーラス入力のアプリケーションから得られる結果のテキストをユーザに戻すようにセットされる。

ユーザは、パラメータ化の間にセットされるパラメータのスーパーセットとしてレガシマネージャにより定義されるパラメータ化されるアトリビュートのリスト1277を特徴付けできる。レガシ133はパラメータ化すべきアトリビュートのリストからパラメータを付加および削除するためのグラフィカルユーザインタフェース1277を提供する。

ユーザは、アンセスタ(ancestor)クラスをレガシするクラス1112としてレガシマネージャにより定義される仮想ルート1269を特徴付けることができる。この手段により、ユーザは、どのスーパークラスシソーラス入力が一つもしくは一グループの部品をレガシするように適用されるかを有効に制御することができる。

第170図は、ユーザが、ステップ615によるレガシ処理、第171図のドロップダウンメニュー選択ファイル1201からオープンを選択するために知識ベースにアクセスする方法を示す。

レジストリサーバ141は、ステップ616でユーザに利用可能な知識ベースのリストおよび権利を質問される。その結果は、部品1203をリトリリーブし、部品1204を編集し、スキーマ1205を編集し、そして部品1206を各知

識ベースがレジストリサーバ141にわかるように示されるようにする権利をも
って、知識ベース名称1202を特微化する選択可能にスクロールされるリス
ト1200としてステップ617においてユーザに表示される。ユーザが、リト
ーバ1203、エディット部品1204およびエディットスキーマ1205の権
利をもち、そして部品の権利1206を否定されることをもって示される例“f
i f i”1202のような知識ベースを選択する時、レガシボタン1207は、
選択された知識ベース1202にユーザがレガシ権をもたなければ、暗くなる。
もし、ユーザが選択れた知識ベース1202にレガシ権を持てば、第172図の
ワークエリア選択ウィンドウ1212を表示して、レガシボタン1207はレガ
シ133で続けるようにステップ618で使用できる。

ステップ619では、ユーザは、ワークエリアを選択およびロックするか若し
くはそれをキャンセルし、ワークエリアを選択しないかのいずれかを行う。第1
72図及び第173図で示すように、クラス階層はステップ624における知識
ベース1216のルートをはじまりとして、表示される。リトリーバ130と同
様にステップ625におけるクラス階層を操作することにより、ワークエリアの
ルートとしてクラス1213を選択し、エリアボタン1214のワークを使い、
ワークエリアをリクエストするかステップ625においてのボタン1215を取
り消す。もし、ユーザがステップ626でテストしたように、リクエストをキャン
セルすると、ワークエリア1217ウィンドウが移動して、最初のレガシーウ
ィンドウ1199が画面に表示される。ユーザがステップ626でテストしたよ
うに、リクエストを取り消さなかった場合は、ステップ627で選択されたクラ
スについて、サブツリーロックがリクエストされる。ステップ628でテストし
たように、もし、ロックができない場合は、エラーダイアログが表示され、ステ
ップ131でリクエストされたサブツリーにおいて、他のユーザが使用している
ことをユーザに知らせる。一方、もしロックされた場合は、その他のレガシー機
能を有するリトリーバ130がステップ629に呼び出される。即ち、それはユー
ーザ1216によって選択されたクラスにルートされたクラス階層である。ステ
ップ630では、コントロールがユーザに復帰する。

レガシー処理機能133は、第174図に示されているパート・スペシフィック

ーション・ウインドウ 1 2 2 4 と共に、A において詳述したごとく、リトリバ 1 3 0 の諸機能を含む。リトリバは以下のような論理を有する。即ち、選択されたクラス 1 2 1 8 におけるサブツリーのアペイラブル・パートが表示され、それはサーチ・クライテリアとしてセットされたいずれのクエリーセクタからも独立した値を有している。アップデートカウンタボタン 1 2 2 0 は、ユーザに対してクエリーのリクエストを明快に行うことができるよう用意されているものであり、これはパートを探すたびにマッチングパートカウントをアップデートすることにより実行される。この機能は、レガシーにおいて別個に備えられている。なぜならばこれは、レガシー知識ベースの処理がパートについて未分類もしくは大まかに分類された部分のいずれかによって始められ、さらにパート階層の高いレベルまで行われ、いくつかのクエリーにとって、パフォーマンスペナルティが可能な結果となっている。サブクラスのためのパートカウントを復帰させるクエリーとは異なったクエリーが実行された場合に、ユーザにコントロールを行わせると意義深い他の効果が達成される。レガシーによって表示されたクラス階層 1 2 2 3 は、その他のアイコン 1 2 2 2 を含み、それらは、他のユーザが利用していることにより、ロックされたクラスを示す。これらのロックは、ユーザの希望により解除することもでき、これには、他のレガシー及びスキーマ編集ユーザのワークエリアを考慮してフィードバックを備え、クラスへのアクセスの際の衝突をより簡単に除去するものである。

レガシー 1 3 3 は、また、第 1 7 5 図に示すようにクラスシソーラス編集のためのユーザインターフェイスを備える。クラスシソーラス編集のための処理は、第 1 7 6 図のフローチャートに示されている通りである。ステップ 6 3 1 では、ユーザはクラス 1 2 2 5 を探し、選択する。そして、マウスの右ボタンを使うことによって表示されるドロップダウンメニューによりシソーラスエントリ編集 1 2 2 6 を選ぶ。ステップ 6 3 2 ではシソーラスリストは、クラス 1 2 2 5 に対するメタデータにより得られるが、これは、ダイナミッククラスマネージャ 1 3 4 を通じて行われる。ステップ 6 3 3 では、ユーザは、リトリバパートウインドウ 1 2 2 8 に戻る前に、第 1 7 7 図に示すシソーラス編集 1 2 2 7 を使い、シソーラスの編集を行う。第 1 7 7 図に例示しているように、シソーラスは、一般に

1つのエントリー1229を有し、これには、1インチピッチマシンのボルト当たり1/4インチ20スレッドを表現するための沢山の常用テキストフォームに適合した一般的な表現を包含する。編集例において、ユーザは同じサイズのマシンのボルト、即ち“1/4インチ”が“.250”もしくはそれについてのパラメントであるシソーラスエントリーを加える。

シソーラス編集は、シソーラス中のテキストストリングのリストを修正することにより行われるが、これは、第178図のフローチャートで示しているシソーラス編集1227によって備えられるコントロールを使うことによってなされる。ユーザは、ステップ635における、7つのシソーラス編集動作のうち、1つを選択する。もし、ステップ636におけるテストで、ユーザがキャンセルボタン1230を選択したことを決定すると、コントロールは、編集が行われたシソーラスをアップデートせずにインボークキング処理ウィンドウに戻る。

OKボタンが選ばれるとスキーマオブジェクトは、ステップ642におけるシソーラス編集のテキストに置き換えられる。追加ボタン1233は、ステップ639において使用され、これは、第179図に示されているカレントで選択されたシソーラスエントリーの下にあるブランチャイン1237を開くためのものである。コピーボタンは、カレントで選択されたステップ637におけるシソーラスエントリー1229の内容を保存するために使用される。これにより、図179に示されているように選択されたシソーラスエントリー1237を置き換えることができる。例示しているとおり、ユーザは、新しいシソーラスエントリー1237の“1/4”1238を図181の“.250”に置き換える。この方法によると、ユーザは、クラスのパーツのパーツのディスクリプションにおいて使われている形式のテキストに適合させるためのパターンをつくるために、シソーラスエントリーを簡単に、再使用することができる。

第182図では、挿入ボタン1234及びシソーラスエントリー1237が選択されているステップ640により、ブランクシソーラスエントリー1240が作成される。シソーラスエントリー1240を取消ボタン1235を使い消去することでステップ641は結果として、第181図で示すようなシソーラスエントリーとなる。

第183図は、クラス階層中のノンリーフクラスにおいて、探すことのできるシソーラスエントリーのタイプを示している。これらのタイプは、微小なサイズであるマシーンボルトの標準的なフォームのテキストディスクリプションで的一部分とマッチする。即ち、このマシーンボルトは“0.25-30=length={0.75inch}L,CAP HD,STLNPHX SKT”といったストリングの一部分を“0.25-20×2.5L,CAP HD,STLNPHX SKT”に変換し、これは、正確なインチの単位によってレングスのパラメータを自動的に引き出す上で非常に簡単でより信頼性の高いものである。第183図は、パートスペシフィケーションウィンドウ1224又はパート編集ウィンドウ1243のいずれかによりアクセスできるクラス階層中のクラスをシソーラドロップダウンメニューから選択することにより、シソーラス編集処理1227を呼び出すことができることを示している。

第184図のフローチャートはエニューメライトされた属性のエニューメレータのためにシソーラスを編集するプロセスを示したものである。第185図に例示しているように、エニューメライト済属性フィニッシュ244が選択されシソーラスエントリーの選択枝を含んだドロップダウンメニューの選択がステップ648として行われる。結果は、第186図に示す通りである。エニューメレータリスト1247は、選択された属性1244のために、取得され、ステップ649において表示される。例として、“cadmium plate”1248といったエニューメレータを選択すると、それに対応したエニューメレータのシソーラスであって第178図のフローチャートに示されている機能であるシソーラスエディタ1227が呼び出される。エニューメレータシソーラスのためのシソーラスエディタについても、エディットパートウィンドウ1243のコラムヘディングから呼び出される。これは、第187図において例として、属性フィニッシュ1244のためのエニューメレータ“Black oxide”1250であるシソーラスエントリー1251が表示され、編集されている。

第203図のフローチャートは、数で表されたもの、テキスト若しくはブールの属性についてのシソーラスエントリーを編集するプロセスを示している。第189図で例示しているように、数値属性としてシソーラス1252、レングスと

して1253が編集される。ユーザは、第190図に示しているパートスペシフィックエディションウィンドウ1224またはパート編集ウィンドウ1243のいずれかからドロップダウンメニュー1246を使うことにより属性を選択する。選択された属性についてのシソーラスは、ステップ656において得られ、第191図に示しているシソーラスエディタ1227によって編集される。この時、ステップ658で、コントロールはユーザに復帰する。シソーラスエントリーの例は、第183図に示すクラスシソーラスエントリー1241によって変換されるマシーンボルトレングスのディスクリプションのいくつかの標準的なフォームに適合する一般的な表現となっている。

第192図では、ユニットファミリーにおけるユニットについてのシソーラスエントリー編集処理を示している。第193図の中の出付したフローチャートでは、ユーザは、ユニットシソーラス編集ボタン1254を、ステップ659におけるレガシーツールのツールバー1255から選択する。全てのユニットファミリーのリストが取り出され、そしてそれらはステップ660でのドロップダウンリスト1256において、ユーザに提示される。ステップ661では、OKボタン1258又はステップ666を通じてキャンセルボタン1259を選ぶことにより、ユニットシソーラス編集から復帰することができる。もし、ユーザが、ドロップダウンリスト1256からユニットファミリーを選択しない場合は、ステップ662は、ユニットファミリー1260についての引き出されたユニットのリストを得る。ユーザは、ステップ663で、引き出されたユニット1260を選択し、ステップ664でユニット1261についてのシソーラスを得る。そして、それは、その後シソーラスエディタ1227によって編集がされる。

第96図は、パート分類レガシー機能1101及びパートパラメータ処理機能1102によってクエリーされたパートについての処理を示している。図195は、パート編集ウィンドウ1243において選択されたパート1262での処理を表したフローチャートである。ステップ667では、ユーザは、属性ディスプレイからパート1262を選択し、ツールメニュー1264からレガシー処理ウィンドウを選ぶ。ステップ668では、レガシー処理ウィンドウは、ユーザのワークスペースのルートからカレントのクエリーのクラスまでの中からクラスパス

を表示する。さらに、選択可能なソースアトリビュート1266および目的地属性1267についてのドロップダウンリストも表示されている。パラメータ処理のための処理可能なターゲットの属性リストが、パラメータセットアップボタン1268が選ばれた場合は、表示される。ユーザは、パート分類1101におけるアンセスタークラスシソーラスエントリーのアプリケーションをコントロールするためにバーチャルルート1269を選択することも可能である。

もしユーザが、レガシー処理ボタン1270を選ぶと、選択した部分は、分類済の1101及びパラメータ処理済の1102の両方となり、これは結果としてパートのパラメータ値がパートディスプレイウインドウ1262に表示されることになる。第196図は、選択した部分1279をレガシー処理した結果を示す。レガシー処理後のパートディスプレイ1281の最初のラインでは、フィニッシュパラメータは“CAD [MIUM PLATE] *”にセットされる。これは、図186において、シソーラスエントリー1249の“CAD [MIUM PLATE] *”に適用させることによるものである。レングス1282は、.5625インチにセットされる。これは、クラス機能、即ち、レングスのための数値属性のシソーラスエントリーおよびユニット“インチ”のためのユニットシソーラスエントリー、におけるクラスシソーラスエントリーのコンビネーションに適用させることによるものである。

ユーザが、分類ボタン1271を選ぶと、選択された部分は分類済パート1101となる。パート分類の結果はパート情報ボタン1273を使うことにより、点検することができる。

ユーザがパラメータセットアップボタン1268を選ぶと、第197図のフローチャートで説明した処理によって、ステップ678における現在のクエリーのクラスの属性が画面に表示される。挿入ボタン1274を選んだことにより、ステップ680は選択された処理可能な属性1276をパラメータ処理された属性1277のリスト中のカレントセクションの上に挿入する。アッドボタン1275を選んだことにより、ステップ681は、選択された処理可能な属性1276をパラメータ処理された属性1277のリスト中のカレントセクションの下に加える。リムーブボタン1278を選んだことにより、ステップ682は、選

扱されたパラメータ処理された属性 1 2 7 7 を移動する。第 1 9 8 図ではパラメータ処理された属性 1 2 7 7 を編集した結果が示されている。

レガシーマネージャ 1 4 5 はダイナミッククラスマネージャ 1 3 4 の構成要素である。これは、ソースアトリビュート 1 2 6 6 及びシソーラスエントリー中のテキストデータの組合せであるクラスへのメタパラメータ、ブーレアン属性、数値属性、ユニット等に基づいた例を自動的にサブ分類し、パラメータ処理する。レガシーマネージャ 1 4 5 による分類はパート分類機能 1 1 0 1 において分類された部分、即ち、ソース属性 1 2 6 6 に対し通常の表現として翻訳されたシソーラスエントリーのマッチングを取り入れたノンバーシングメソッドによって達成される。好結果の組合せの各々は、マッチングクラス、属性、又はエニュメレータシソーラスエントリーを探すことのできるクラスのスコアを増大させる。シソーラスエントリーの組合せはクラス階層へ再帰的に実行される。これはパートインスタンスがカレント定義したクラスを始めとして、再帰的呼出し元が復帰することでシングルエリミネーショントーナメントの方式でスコアを比較することにより行われる。もし同系クラスの中で明らかなウィナーがあると、そのウィナーは再帰的呼出し元リターンに引き継がれ次のレベルで対比されることとなる。同系グループ中の最新のウィナーと同じスコアでクラスが達成された場合は、ウィニングスコアは保存され、最新のウィナーは対比済としてマークされ同系クラスグループのウィナーは宣言されない。しかしながら、対比済マークがされたウィナーのスコアに対して優位のマッチングスコアを達成した同系クラスは同系グループのウィナーとして宣言されることとなる。同系グループ中にウィナーがいない場合は、そのグループの中のスーパークラスがウィナーとして宣言され次のラウンドの対比で、そのウィナーの同系と対比することとなる。クラス階層の再帰的系統が完了されると、もしウィナーがサブツリーのクラスの中から選ばれていれば分類されたパートインスタンスはウィナークラスにセットされたオーナーを有することになる。そしてパートインスタンスは再度分析される。即ち、まず最初に、そのオーナーの間にシソーラスエントリーを有し、そして分類パート機能 1 1 0 1 を呼び出し、ソース属性 1 2 6 6 によって定義されたテキストパラメータによりアブライされたところのクラス階層のルーツ若しく

はバーチャルルーツクラスのいずれかを備えることによって行われる。これらの結果はテキストの取り除かれた一部分として残ることになる。この修正されたテキストは、目的地属性1267によって定義されたテキストパラメータをセットするために使用され、さらにパートインスタンスを分類するために使われたシソーラスエントリーマッチの組合わせに関するフィードバックをユーザに提供する。

第133図に戻ると、レガシーマネージャーの自動分類機能部は、ユーザによって選ばれたソース及び目的地属性がテキスト属性であって、さらにパートインスタンスのオーナークラスについてローカルか若しくはステップ1104でそのクラスで承継したものであるかのいずれかであることを確かめることにより開始する。もし、違反の属性がステップ1105において検索されるとパートインスタンスはその分類を変更せずに元へ戻る。さもなければ、ソース属性テキストパラメータのローカルコピーを作成すること、及びステップ1106でのレガシー属性をイニシャライズすることにより、イニシャライズが実行される。このローカルコピーは、分類トーナメントが進行するにつれクラスツリーのそれぞれのノードにおけるクラスのスコアを含んでいる。インスタンスは次にステップ1107において分類され、前述した概要のメソッドへと続く。もしステップ1107のサブツリーを通じてのシソーラスエントリーのアプリケーションにより、残ったワーキングストリングがステップ1108において長さが0であると決定されると、目的地属性パラメータはステップ1109のパートインスタンスについては定義があたえられないようにセットされる。さもなければ目的地属性パラメータはステップ1110のワーキングストリングの値にセットされる。分類されたインスタンスはステップ1111の呼出し元に戻る。

第134図では、ステップ1107で言及したパートについての分類を行うためのメソッドが示されている。ステップ1112では、現在のクラスのアンセスターがレガシー処理される。これはユーザによって準備されたルートクラス若しくはバーチャルルートクラスから順番にシソーラスエントリーがアプライされることによる。次に、インスタンスのオーナークラスのサブツリーは、ステップ1113でソースアトリビュート1266について全体的に一番良いマッチを提供

するクラスを探す目的でインスタンスをレガシー処理するために再帰的に受け継がれる。もしウィナークラスがステップ 1114 において決定されたパートについての現在のオーナークラスと異なる場合は、ステップ 1116 で分類されたパートインスタンスが復帰された後に、クラスのオーナーはステップ 1115 においてウィナークラスにセットされる。

第 135 図では、ルートからオーナークラスのパートへ注文されたリストを作成するルート又はバーチャルルートクラスを含み、パートのオーナークラスからのクラス階層を上昇させるステップ 1113 と共にレガシー処理されたアンセクタークラスが示されている。このリストの最初のクラスであるルート又はバーチャルルートクラスはステップ 1114 において得られる。もしステップ 1114 が、ステップ 1115 で決定したとおりに処理するクラスをうまく取得すると、そのクラスのためのクラスシソーラスエントリーはステップ 1116 において処理される。この処理は、ワーキングストリングに対する修正という結果となりうる。ステップ 1117 では、リスト中の次のクラスが取得され、その制御は次にステップ 1115 にもどり、リストの各クラスを処理するループを備える。クラスリストの最後までいくことにより、ループが終結したときは、アップデートされたスコアと修正されたワーキングストリングを持つレガシーアトリビュートは呼出し元へ戻る。

第 136 図では、サブツリーの中で最良のマッチングクラスを決定するためのインスタンスをレガシー処理するメソッドが示されている。まず最初に、ステップ 1117 でワーキングストリングの長さが 0 であるかどうかチェックする。ステップ 1118 では、ウィナークラスを現在のクラスにセットされ、マッピングクラスにおいてさらなる選択をする上で影響を与えることのできる文字のいずれもがワーキングストリングに含まれていないとなった場合には戻ることとなる。もしワーキングストリングが 1 又はそれ以上の文字を有する場合には、現在のクラスにおけるシソーラスエントリーがステップ 1119 において処理される。ステップ 1119 でアプライされたシソーラスエントリーのいずれもがワーキングストリングとマッチしなかった場合には、クラスタイプがステップ 1122 においてテストされる。クラスタイプがプライマリーとしてマークされ、オーナーク

ラスのインスタンスが、ステップ1123でテストしたように現在のクラスと同じでなかった場合は、ステップ1125まで処理が進行し、現在のクラスはこのサブツリーのウィナーとして戻ることとなる。これらの処理は、プライマリークラスが少なくともひとつのマッチングシソーラスエントリーを有し、不必要な処理を防がない限りは、プライマリークラスのサブツリーが受け継がれないことを確かめるために行われる。クラスタイプがコレクションとしてマークされた場合は、シソーラスエントリーはステップ1124-ワーキングストリングのパートについてマッチ及び削除することを意図する0シソーラスエントリーのタイプをもつコレクションクラスはより良いマッチを探するため、いつもサブクラスを備えている。もし探せなかった場合は、サブツリーのウィナーと共に現在のクラスが宣言された状態で処理はステップ1125まで進行する。セコンダリーとしてマークされたクラスは、いずれかのシソーラスエントリーがワーキングストリングとマッチしたか否かに限らず下位へ受け継がれる。

サブツリーの下位段階への受け継ぎが前述のルールに従ってなされた場合、現在のクラスのローカルアトリビュートのシソーラスエントリーはステップ1126において処理される。ステップ1127では、現在のクラスに定着したサブツリーを通じてトーナメントのウィナーとなったものは、一時的に現在のクラスとして宣言される。その後第137図でのステップ1128を経て処理が続き、ここではサブツリーはより良いマッチを探求する。これは、サブツリーの再帰的受け継ぎによって達成され、これはまずステップ1129における現在のクラスのためのサブクラスのリストを取得することにより始められる。ステップ1130でテストしたようにこのリストが完全に処理されていない限りは、ステップ1131でリストの次のクラスが取得され、レガシー処理されたインスタンス機能1113が再帰的にコールされるといった内容のループは実行される。この機能はいつもウィナークラスセットと一緒に戻るが、これはちょうどサブツリーの中から最良のマッチが探せなかった場合の現在のクラスと同様のことといえる。前記のように戻ったウィナークラスのスコアはステップ1114において現在のウィナーと比較される。第1クラスのマッチのカウン트에最も重み付けがされ、そのあと第2クラス、コレクションクラス、非数値的属性マッチ、数値的属性マッ

チ、へと続く。そして最終的にタイプブレーカーとして、処理後、より短いワーキングストリングを有するクラスが採用される。もし復帰したウィナークラスが現在のウィナーより低いスコアの場合は、ウィナークラスは拒絶され、ルーブはステップ 1130 を続行する。戻ったウィナークラスが現在のクラスと同じであった場合は、現在のウィナーはステップ 1115 で対比済のマークをつけられこのサブツリーにおいてウィナーとして宣言される。これはすべてのサブクラスにおける対等なマッチについて、最初に処理されたサブクラスを優遇するといったことを防ぐためである。戻ったウィナーが現在のウィナーよりも高いスコアを持った場合は、ステップ 1116 において、戻ったウィナーが現在のウィナーとして保存され、以降のこのサブツリーにおける競争は新しいウィナーのクラスにおいて行われる。

サブクラスのリストがステップ 1130 でのテストによって示されたようにその処理が失敗となった場合は、処理はステップ 1125 まで続行される。第 138 図では処理済マークがされたウィナーのフラッグがステップ 1186 でテストされる。即ち、フラッグがセットされるとカレントウィナーは拒絶され、カレントクラスがステップ 1187 におけるサブツリーのウィナーとして宣言される。ステップ 1188 でテストされたようにいずれかの処理によって新しいウィナーが宣言される結果となった場合は、カレントワーキングストリングはステップ 119 のサブツリーにおける対比から復帰したワーキングストリングに置き換えられる。カレントスコアはステップ 1190 において、新しいウィナーのスコアに更新され、このカレントスコアはトーナメントの次のレベルのクラスにおいて対比するために使われる。新しいウィナーが宣言されか否かにかかわらず、サブツリーにおける最終のウィナークラスはステップ 1121 の呼出し元に戻る。

第 139 図では、分類する目的でなされる属性の処理が示されており、これはステップ 1131 のクラスにおけるローカル属性のリストを得ることから始まる。これらの属性はすべてステップ 1132 でのテストにより制御されたループによって処理され、ステップ 1133 では分析のため、ローカル属性のリストの中から新しい属性を得ることとなる。属性のワーキングストリングへのシソーラスエントリにおけるマッチングの詳細は、ステップ 1134 での属性タイプのテ

ストとして制御される。エニユメレートされた属性はそれ自体ではシソーラスエントリーを有していない。パートデスク립ションにおいて探すことができるであろうエニユメレートされた属性と連結された用語は属性の可能値、即ちエニユメラータである。各エニユメラータはシソーラスエントリーのリストを持ち、それらのいずれかはワーキングストリングの一部分とマッチすることとなるものである。各エニユメラータをテストするためには、ステップ 1 1 3 5 においてエニユメラータのリストが作成される。ステップ 1 1 3 6 で他のエニユメラータのテストが成功している限りは、シソーラスエントリーのリストによって構成されたシソーラスがステップ 1 1 3 7 より得られ、ステップ 1 1 3 8 のワーキングストリングに対して処理がされる。シソーラスにおけるエントリーがマッチすると、ステップ 1 1 4 0 のクラスにおいて、ワーキングストリングに対するマッチの段階を向上させながら非数値属性のスコアが増加する。シソーラスエントリーのいずれもがマッチしない場合は、ループは続けられ、各エニユメラータを順番に処理する。

属性がテキストストリング又はブールの場合には、その属性は、より先の処理に進む前にどのようにしてローカルのワーキングストリングを変換するか、また、ワーキングストリングでマッチされた場合に、どのパターンがテキストストリング又はブールの属性のマッチを構成するかといったことの両方を表現するシソーラスエントリーを備えたシソーラスを有することがある。このシソーラスはステップ 1 1 4 1 で取得され、あらゆるスキーマエレメントについてのあらゆるシソーラスを処理する通常の機能を使いステップ 1 1 3 8 で処理される。シソーラスエントリーがマッチすると、非数値属性スコアがステップ 1 1 4 0 でのクラスにおいて増加される。

数値属性は、数値属性それ自体及びあらゆる連結ユニットシソーラスといった両方の観点から評価されなければならない。数値シソーラスエントリーは数字と他の数値シンボルとの組み合わせのマッチと同時にテキストストリング中から数値情報を探すための適切なコンテキストを示すパターンのマッチを行うことを試みる。ベースのためのユニットシソーラス及び属性におけるユニットファミリーでの引き出されたユニットはワーキングストリングにおいて探すことのできる異

なったコンバーチブルユニットの全体にわたって差別するパターンを備える。例えば、そのパターンの例として、インチ(inch)については、“in”又はフィート(feet)について“ft”等で、こうした情報はレガシーマネージャーによって正しく翻訳され変換される。数値属性のシソーラスはステップ1141で得られステップ1138において処理される。ステップ1139でテストされたように、いずれのシソーラスエントリーもがマッチされなかった場合には、ステップ1142で、数値属性におけるユニットファミリーのユニットのリストが得られない限りそれ以後の処理はなされない。

ステップ1143のリストにおける、他のユニットのためのテストによって制御されているループは、ステップ1144で次のユニットのためのシソーラスを得て、ステップ1138のワーキングトリリングに対してそのシソーラスを処理する。そして、ステップ1139でのマッチのためのテストを行う。マッチは、正しい組み合わせの数とユニットがワーキングストリングにあるか否かを表示しながら、ステップ1145のクラスの数値属性スコアを増加させる。

第140図はシソーラスが定義されることのあるすべてのタイプのスキーマオブジェクト例えば、クラス、数値属性、エニューメレータ、ブーリアン属性及びテキスト属性-のためのワーキングストリングに対するシソーラスの処理についての一一般化したメカニズムを示している。

このメカニズムは、シソーラスエントリーの成功したマッチがワーキングストリングの修正としてのこされるべきか否かについて表示するためにフラッグと共に呼び出される。処理はすべてのストリング又はシソーラスエントリーのリストを得ることによりなされ、これはステップ1146の呼出し元によって提供されるシソーラスを作り上げる。これらのエントリーはループで処理され、これはステップ1157のリストにおける他のシソーラスエントリーのためのテストと、ステップ1157における成功パターンマッチのテストの両方によって制御されるものである。ループ内の反復ごとに次のシソーラスエントリーがワーキングストリングに対して処理される。

ワーキングストリング中のテキストをマッチさせるためには2つの一般的なタイプのシソーラスエントリーがある。まず第1に、修正又は編集シソーラスエ

ントリーと呼ばれるもので“v/”又は“g/”のいずれかで始まる。これらのシンーラスエントリーは本質的にはUNIX viエディタの中の編集コマンドと同じである。例えば、シンーラスエントリー“g/x.*([0-9.])[0-9.\\][0-9.\\/*)/s/length={\\1 inch}”は最初のスラッシュのペアの間の一般的な表現パターンにマッチしたストリングにおいてのみ作動し、そのストリングにおけるテキスト“length=”が置き換えられ、続いて前記表現パターン（括弧の間にしめされているもの）において獲得された部分にマッチしたワーキングストリングの中のテキスト、そして“inch”という語へと置き換えられる。例えば、ワーキングストリング“1/4-20 x 1.25”は残った話ワーキングストリング“1/4-20 length={1.25 inch}”を作りだす。このように、最大限に通常の表現を採用しワーキングストリングの一部を獲得・再利用する能力を持ったシンーラスエントリーのタイプは、特別のクラス又は属性によく見られる標準フォームのテキスト断片についてすべてのテキストが正確で規範的なフォームに従っていることを要求することなく信頼性の高い評価をもたらす。その結果通常レガシーパーツデータソースにおいて探すことの出来る広い範囲にわたるデータの種類の成功裡に開発することをレガシーに可能ならしめるものである。

“v/”のフォームをもつタイプのシンーラスエントリーは、ワーキングストリングにおいて通常の表現が含まれるシンーラス中でマッチされていない場合はファイラー又は最初のパターンがマッチされるとすることを除いては“g/”フォームのものと同一に働く。このことは、存在しないデータであると発見された場合においても、データが提供されれば後のシンーラスエントリーにおいてよりシンプルな処理を行わせるワーキングストリングの選択的な修正を可能とするものである。

編集スタイルのシンーラスエントリーにおける処理は、ステップ1148で最初のスラッシュのペアの間のファイラーを取り出すことにより始められる。マッチすべきパターンはそのあとステップ1149で取り出される。パターンがない場合は上記の例のごとくそのパターンはファイラーと同一としてデフォルトされる。フラグはステップ1151でシンーラス自身によって定義されたトラ

スフォーメーションとは別のワーキングストリングにおける後の修正を防ぐためにセットされる。ファイラーがワーキングストリングとマッチすると、ステップ 1152 でテストされるようにファイラーは “v/” シソーラスエントリーがもつとは反対の意味を念頭におき、そして前記パターンについては、修正フラッグがセットされステップ 1157 でのワーキングストリングに対してのテストがされる。このパターンがマッチし、ステップ 1158 で呼出し元がワーキングストリングを修正することを要求して修正フラッグがセットされると、ワーキングストリングの中の、パターンにマッチしたテキストはステップ 1159 で獲得されたテキストの適切な拡張がされた交換テキストによって交換される。シソーラスエントリーがマッチすると、ステップ 1160 においてブーリアン値の真が呼出し元に戻される。

編集されていない若しくはシンプルなシソーラスエントリーはマッチする傾向にあり、選択的にワーキングストリングからマッチされたテキストを取り除く結果となる。これらは “v/” 又は “g/” から始まらないことで区別される。パターンはステップ 1154 でシソーラスエントリーにセットされ、フラッグはステップ 1155 でワーキングストリングを修正するためにセットされる。交換テキストは、後のシソーラスエントリーで検出でき、ステップ 1156 において単一の文字 “!” にセットされる。そしてこの交換テキストは目的地属性 1267 又はシソーラスエントリーマッチング “!” についての点検のいずれかによって決定づけられる簡単な手段を提供するものである。このような方法で、シソーラスエントリーをマッチングする効果が、前の段階でのシソーラスエントリーの成功したアプリケーションにおいて条件付きで得られる。こうした観点で、編集シソーラスエントリーの処理は続けられる。より詳細には、ステップ 1157 のワーキングストリングでのパターンがマッチしているかのテストを持ち、その後ステップ 1158 のテストによって制御されステップ 1159 でカレント交換テキストとして定義される文字 “!” によってマッチされたテキストの選択的な交換がなされる。

ステップ 1147 のテストによって表示されたマッチが不可とされることなくシソーラスエントリーのあらゆるリストが処理された場合は、ブーリアン値の偽

がいずれのマッチも生じなかったことを表示するために復帰する。

第141図では、クラスについてのシソーラスの処理が示されており、これについては、ステップ1116においてクラスがコレクションクラスか否かを最初に決定する。コレクションクラスはシソーラスを備えることがあり、これはマッチがされる範囲を制限することによりパフォーマンスを調節し又は信頼性を向上させるためにサブツリーの下降を防ぐものである。しかしながら、編集シソーラスエントリがたえずワーキングストリングの修正の結果となるのに対して、シンプルなシソーラスエントリはコレクションクラスにはあまり使われない。このためステップ1164のワーキングストリングの処理を防ぐためフラグがセットされる。逆に、第1又は第2段階のクラスでは、シンプルなシソーラスエントリマッチは、いつもワーキングストリングからマッチされたパターンを取り除く結果とされなければならない、またそれゆえにストリングを処理するするためのフラグがステップ1163でセットされる。このいずれかの場合においてもスキーマオブジェクトのためのシソーラスを処理する機能はステップ1165のクラスのために呼び出され、結果はステップ1166へ復帰されることとなる。

両方のクラスをマッチさせることにより自動的にサブ分類すること、及びサブツリーにおけるシソーラスエントリを、ソース属性1266によって定義されたテキストパラメータヘアトリビュートすることの他に、一又はそれ以上の指定された属性がシソーラスエントリを使うことにより自動的にパラメータ処理される。

第142図では、102のパラメータ処理された部分が示されている。1101の分類された部分のように、102はソースと目的地属性がローカルか又はステップ1104においてパラメータ処理されたインスタンスのオーナークラスへ承継することができるかをチェックしながら処理を開始する。ステップ1105において違反した属性を検出した場合は、インスタンスは単にステップ1169に戻る。ソース属性のためのテキストパラメータはワーキングストリングへとコピーされ、レガシー属性はステップ1106でイニシャライズされる。インスタンスのオーナークラスとルートのクラスの間におけるシソーラスエントリの何らかの効果を保存するためには、レガシー処理されたアンセスタークラスがステ

ップ1112で実行され、オーナークラスのためのシソーラスエントリーについてもステップ1119において処理される。非数値的パラメータは先ず最初にステップ1167でレガシー処理され、その後ステップ1168で数値的パラメータが処理される。こうした順序はシソーラスエントリーのコレクションの定義をより簡単にすると同時に、エnumレーター若しくはその他の非数値的属性においてしばしば見受けられる数値的シソーラスエントリーと数値的データの間に生じる衝突を少なくすることを可能にする。例えば、セラミックキャパシター誘導体“X7R”は簡単な数値的シソーラスエントリーを適用することにより偶発的に省略されることがある数字を含むが、これは非数値的属性を先に処理させることにより守ることができる。すべてのパラメータがレガシー処理されると、パラメータ処理されたインスタンスはステップ1169へ戻る。

非数値的パラメータをレガシー処理するためには、第143図で示すステップ1167の処理が行われる。ステップ1170では、承継されたすべてのリストと、パートインスタンスのオーナークラスのためのローカル非数値的属性が取得される。このリストはさらに、第194図のダイログ1263を使用したユーザによって選択された一定のターゲット属性に制限されることになる。このリストにおける反復は、第145図のステップ1132により、リストの他の属性のためのテストによって制御されている。そしてステップ1139において起こりうる好結果となったシソーラスエントリーがマッチすることになる。このループにおいては、リストの次の属性がステップ1133より取得され、その属性のタイプがステップ1134で決定される。

属性タイプがエミュレートされると、各エnumレータのためにシソーラスエントリーを評価するという過程が同じに実行されるが、これは第139図で示したのと同じものである。即ち、エnumレータのリストがステップ1135で作成され、ステップ1136でテストされたようにリストが空になるまでループされる。各エnumレータには、ステップ1137でシソーラスが取得され、ステップ1138で処理される。シソーラスエントリーがステップ1139のようにマッチすると、属性のためにエnumレータされたパラメータが、シソーラスエントリーがマッチしたエnumレータへセットされる。しかしながら、パラメ

ータは現在定義されなかった場合にのみセットされる。現在ある値にセットされたパラメータは以前に行われた、より信頼性の高い処理によってセットされたと推定される。それは即ち、直接的に導入されたデータ又はリトリーブ若しくはレガシーインターフェイスの部分編集カーパビリティを通じて人間により値が入れられたもののいずれかである。

同様のメソッドはステップ 1134 のテストで検出されているようにテキストストリング及びブール属性に使われている。属性のためのシソーラスはステップ 1141 で取得され、シソーラスはステップ 1138 のワーキングストリングに対して処理される。シソーラスエントリがマッチし、パラメータがカレント定義されない場合はシソーラスは適切にセットされる。即ち、ブールにおいては真となり、又はストリングがアトリビュートされた場合はマッチされた結果のいずれかとなる。

第 145 図ではステップ 1168 における数値的パラメータのレガシー処理が示されている。これは、承継されたリスト及びローカル属性を集合させステップ 1175 中のスキーマにおいてこれらについて定義された順序に従うことにより始められる。リストは第 194 図のダイアログ 1263 を使いユーザにより選択された一定のターゲット属性にさらに限定されることになる。処理されるべき属性が残っている間、ステップ 1133 では次の属性が取得され、そのシソーラスはステップ 1141 においてそのメタパラメータから導き出される。シソーラスエントリがマッチすると、属性におけるユニットファミリーのユニットのリストがステップ 1142 において作成される。各ユニットはテスト 1143 により制御されたループにより処理される。即ち、ステップ 1176 において取得されたユニット及びステップ 1144 において取得されたユニットシソーラスはステップ 1138 において処理がなされる。ステップ 1139 のテストによって指示されたマッチングユニットシソーラスエントリはステップ 1177 において、結合された数値的パラメータをセットする結果となる。非数値的属性の場合では、現在セットされたパラメータ値はこのステップ中においては上書きされない。数値的属性のリストが空になると、インスタンスはステップ 1178 に戻る。

レガシー処理機能 133 はまた分類プログラム 3001 及びスキーマジェネレ

ータを包含する。

第200図で示している分類プログラムは、正式なオブジェクト名と人間が入力し、省略、スペルミスを含んでいるテキストの記述とをマッチさせるために使われる。

分類プログラムの目的は2154のスキーマオブジェクト名から蓄積された知識（クラス名、属性名、ユニットファミリー他）を、新しいパートを位置づけることがりまたは2153で取得された与えられたディスクリプションによってパートのセットを見にいくことがあるオブジェクトに対しローケーションを提案することである。分類プログラムは、スキーマの中のポテンシャルローケーションのセットでありパートディスクリプションが2157において分類されたものであるアウトプットを、生成する。ポテンシャルローケーションのセットは2158で人間によって見直される。選ばれた分類はその後と2159のインポートマップに置かれることになる。

分類プログラムは2つのワードマッチングテクニックを使うことにより作動する。1つのマッチングテクニックは“ビッケルアルゴリズム(Bickel Algorithm)”と呼ばれるものであり、他のマッチングテクニックは“サウンデックスアルゴリズム(Soundex)”と呼ばれるものである。これらのアルゴリズムはターゲットとなる語についての候補語のマッチを捜し当てる上で異なったアプローチをとる。ビッケルアルゴリズムでは、マスクはターゲット語に共通の文字を表しており、ターゲット語は各文字の使用頻度を基にして値付けされる。スコアの総計が高ければ高いほどマッチがうまく行われる。ビッケルアルゴリズムはアートについて熟練した人々に広く知られているところである。

サウンデックスアルゴリズムでは、ターゲット及び候補となる語の両方について使用され文字の音を表すマスクが、的確なマッチをするため又はマスクにおいて一定のローケーションにまでマッチアップするためにチェックを行う。サウンデックスアルゴリズムについてもアートについて熟練した人々に広く知られるところであり、詳細には表されることはない。

分類プログラムはすべてのスキーマオブジェクト名を、第146図におけるステップ925のオブジェクト指向データベースより引き出す。スキーマオブジェ

クト名が引き出されると、926において正確な語に分割される。ビッケルアルゴリズムを使い正確な語のそれぞれに代表的なフォームへ符号化すると、927においてサウンデックスアルゴリズムがあらわれる。ステップ925から928までの処理はすべてのスキーマオブジェクト名が引き出されるまで続けられ、正確な語に分割され、符号化される。925において実行される引出し、分割、そして符号といった処理の他に、分類プログラムはスキーマツリー構造のどこで各語が使われたかを記憶する。これはツリーの異なったロケーションにおいて同じ語が使用されることがあるため、どの語がどこで使われたかを記憶しておくのは重要なことである。

ユーザ入力ステップ929において得られる。ユーザ入力は、カスタマーから得られたデータであり、925から928のすべてにおいて得られるスキーマ名データを使って分類される部分をあらわす。ユーザデータはテキストデータであり、このデータはスキーマ語に対してマッチングを行う目的で正確な語に分解される。ステップ930はユーザ入力ストリングを正確な語に分解する。これらの正確な語は目的語についての略語となり、スペルミスを含んだり他の面で不良のフォームとなり得るものである。

スキーマ930のユーザディスクリプションから得られた語はその後、第147図中のステップ931におけるビッケルアルゴリズムとサウンデックスアルゴリズムによって符号化される。ビッケルチャクターマスクは、ステップ932から935においてどのスキーマ語が最適な候補マッチであるかを決定するためにアプライされる。ステップ932ではスキーマ語を選ぶ。ステップ933ではビッケルアルゴリズムによって与えられたスコアについてそれが最高のスコアマッチであるかどうかを決定するためにテストを行う。もしそれが最高のスコアであればその結果は934においてポテンシャルマッチのリストに保存される。サーチは935において、マッチですべてのスキーマ語が検査されるまで続けられる。

第148図のステップ936は前のサーチループの結果を検査し、934でポテンシャルマッチについて作成されたリストに見られる932から935のルールにより、最高のスコアと的確にマッチする語を選択する。これら927で作ら

れたサウンドデックスマスクは、937の元の入力語と洗練された候補のセットがどのくらいうまくマッチされているかをテストするために使われる。もし語がサウンドデックステキストから外れると、その語は938での候補リストから外される。937でサウンドデックステキストが成功すると、そこでの語は939でそれ以降の使用のために保留され、他に残った候補語がないかどうかを決定しサーチは940へと継続される。940では、候補リストの中にまだ語があれば、936でプログラムは続行される。その他の場合はプログラムは941において続けられる。

この時点で、サーチツールとして役立つための選択肢は依然として多すぎるほど存在することもあり、またこれまでの説明においては単一の語についての場合のみにしか触れていない。多くのケースではパートのディスクリプションには複数の語が含まれている。各語は独自の意味を持つが、集合的な各語のセットの文脈においては、各語それ自体が意味するものよりも多くの意味を含んでいる。言い換えれば、各語のグループは、それを構成する各語としてのパーツの合計よりも、より意味深いものであるということである。このことは、各候補語がクラス構造のどこからきたものかを記憶することに活用される。前述のマッチングテクニックにより探した語のアンセスター（系統）を比較することで、コモンスレッド、即ちツリーを通じて他のスレッドよりも多く使われているもの、を探することができる。このようなポピュラーなスレッドは、ツリー中で、与えられたパートディスクリプションが探せるか若しくは位置づけることのできる好ましい場所をあらわす。このように、共通のアンセスターのスレッドを探すための処理が941において示されている。第149図のステップ942ではこのプロセスの処理をみて、アンセスターのコンビネーションにおいて全くストリングスを得られなかった語があるかないかを決定する。全くストリングスを得られなかった語はそれが唯一のものでない限り、放棄される。（言い換えれば、共通のアンセスターが探せなかったということである。）

以上のアルゴリズムのアウトプットはその後ようやくステップ943で示すように、テキストベースインターフェイス又はグラフィカルユーザインターフェイスを通じてユーザに提示される。こうしてユーザは自分が何を探しているかを小

さなセットの中から選択することができる。この処理は対話式、又はパッチインターフェイスのいずれかを通じて分類プログラムへ提示される各ディスクリプションについて繰り返される。

スキーマジェネレータは、省略及びスペルミスを含んだ、人間によって入力されたテキストディスクリプションをオブジェクト指向データベースへジェネレートするために使われる。スキーマジェネレーションプログラムには3つの目的がある。第1は、人間によってジェネレートされたテキストディスクリプションからオブジェクト指向データベースのためのスキーマクラス構造をジェネレートすることである。第2は、ジェネレートされたクラスのそれぞれについてのクラス密度を決定することである。第3は、現存するデータをさらにボリュートし、追加のバリエーションクラスを作ること防ぐためにインプットデータの誤ったスペルを組み合わせることである。

第199図に示されているスキーマジェネレーションツール3002は、インプットとしてカスタマー又はユーザパートディスクリプションを使いスキーマを作るために使われる。このパートディスクリプションデータは2150で人間により入力されることから、データはミススペル、タイプミスを含む傾向にある。スキーマジェネレーターはユーザディスクリプションを2151のスキーマ構造へと下げる。アウトプットはスキーマ構造とパートディスクリプションのいくつが2152における各スキーマクラスに置かれるかを表示するパートマッピング密度である。

スキーマジェネレータは第150図におけるステップ960の不定パートのディスクリプションを読み込む。ディスクリプションが得られるとステップ962は963へその処理を続けることを実行する。インプットが見つけれない場合はステップ962はアウトプットをディスクファイルへプリントし、処理を終了させる。

不定パートは963における語に分解される。各語は964で示されたレベルに存在する他のすべての語に対して比較がされるが、これは語が前記レベルにおけるリストに追加する必要があるかないかを決定するためである。カレント語が以前に使用したいずれの語ともマッチできなかった場合は、プログラムは前記レ

ベルにおける内部的な語リストに語を追加するために第151図の971へ進む。

どのようにマッチが行われたかに関係なく（新しい語を追加したり、存在する語のマッチを行うなどによって）ストリング中の新しい語は再帰的方式によって、マッチされた語の従属語に対して検査がされる。この処理はインプットが空になるまで続けらる。即ち各ディスクリプション及びすべてのディスクリプションが空になるまでの両方の場合に続けられることになる。

マッチングに使われるテクニックは第150図のステップ964におけるビッケルアルゴリズム、ステップ966のタイプミスマッチャー、そしてステップ966における省略語マッチャーである。ビッケルアルゴリズムではマスクはターゲット語に共通する文字を表し、各文字の使用頻度に基づいて候補語に値付けがされる。スコアの総計が高ければ高いほどマッチはうまくいく。ビッケルアルゴリズムはポテンシャルマッチの最も主要なセットを探すために使われ、第150図では964において使われている。

人間はパートディスクリプションをエンターする上で個々で首尾一貫していないことから、ビッケルアルゴリズムによりサーチのアウトプットを洗練する別のテクニックが採用する必要がある。いくつかのケースでは、人間はフルワードのディスクリプションについて省略語を使っている。省略語は一般的には語を短くするか、語の中から文字を消すことにより作成される。

ステップ966では、省略語マッチャーは、比較対照においてエラーに遭遇した場合はターゲット語を引き伸ばすことを試みる。比較対照が失敗に終わるたびにターゲット語は1文字引き伸ばされる。こうして比較対照は次の文字へと移り再開する。もしターゲット語が候補語よりも長くなった場合は、比較対照は失敗することになる。もしターゲット語が候補語との比較対照を完了する前になくなってしまう場合は、推測された省略語によって一定の率の語がカバーされたときはそれを宣言する。求められるカバー率は調整可能であり、各データセットにおいて調節される。以下にいくつかのサンプルを記す。

B l t —→ B * l t

B o l t B o l t 比較対照が100%カバーしている場合
 R g s t r → R * g * s t * r
 R e g i s t e r R e g i s t e r
 比較対照が100%カバーしている場合

M i c r o p r o c M i c r o p r o c
 M i c r o p r o c e s s o r M i c r o p r o s s o r
 比較対照が75%カバーしている場合

*印の文字はストリングが引き伸ばされた場所に挿入されたことを表している。
 文字の値は処理とは無関係である。従ってどの文字を使用してもよい。ステップ966の結果が1つのマッチのみを生むものであった場合はステップ967のスキーマジェネレータは、ステップ970で使用中の語とカレント語とを組み合わすことを判断する。

ステップ968でのタイプエラーマッチャーは、同じ語と思われる語を組み合わすことに使われ、これは、同じ語により、人間によって検索することができるがコンピュータはこれを行うことができない。

タイプミスは、人間が通常の“クワージー(quarterly)”キーボードによってデータを入力する場合に、意図したキーをはずしてそのかわりに隣接したキーを打ってしまうことにより起こる。例えば“Adhesive”という語を例にとってみると以下ようになる。

A d h e s i v e → 意図した語
 A s h e s i v e → dとすべきところsとした場合
 A d n e s i v e → hとすべきところnとした場合

上記における2つのタイプミスでは意図したキーは、使用された文字のキーに対して物理的に隣接したところに探すことができる。

タイプエラーを含んだ語をマッチさせるためには、2つのことを行う必要がある。まず第1に、キーボード上の各キーについての隣接キーのすべてをマップにすることである。第2に、タイプエラーを含んでいると推測される語を比較する場合には、比較対照の処理において、マッチしなかったターゲット語の文字の隣

接キーを見ていくことが必要となる。もしキーがキーマップ上の隣接キーとして探せた場合には比較対照の処理が続けられる。それぞれのエラーはカウントされ、最後にはマッチにおいて特定の数エラーを数え上げることとなる。もしいくつかの候補語がまだ存在する場合は、その中で最もエラーの少ないものが選択される。

これまで説明したタイプエラーの他に、置き代わった文字を含むという特別の場合もある。この場合には、これまで説明したメソッドによっては検出することができない。しかしながら、文字対文字ベースでの絞り込みをおこない、0ではないいずれかの数の絶対値をとることで、同じ非ゼロ値を有する隣接文字によって置き代わった文字を検出することができる。文字の置き代わりはエラー記録においてエラーとしてカウントされることはない。“Positive”という語を以下で例にとる。

P o s i t i v e

P s o i t i v e

隣接キーを使ったマッチング処理によって組み合わせがされた場合、P o s i t i v e という語は以下のようにマッチされることがある。

P o s i t i v e

P s o i t i c e

図151のステップ968においてタイポ・マッチング・プロセスの結果がシングルマッチとなったときには、ステップ968において現在のパート記述とステップ970における現在のパート記述とを結合させる決定がされる。

これらの手法の組み合わせによりマッチングが推定されると、次の問題はワードの綴りの訂正の選定である。新しい綴りが見つけれられかつ特別の立候補ワードがマッチする度毎に、誤った綴りのものはステップ970において立候補したものと組合わされる。立候補したワードが、置換が完了した際に何らかの作用が行なわれた理解ある対象として取り扱われたときには、与えられた綴りが認められた回数が記録される。上記対象は最も普遍的な綴りとして現在の状態を招来することができる。人は大抵の場合正確に綴ろうとする傾向があるから、誤った綴りが

マッチせず、カウントされない等の場合には、それらが正確に綴られたワードになるようにする。綴りを照合された時の上記対象は、最も実用的な綴りに対応している。

ワードが特殊なユーザーにより部分記述において常に戻って綴られた場合には、すべての誤った綴りは最も一般的なもの、すなわちその場の対象におけるクラス実体として示唆されたものに集まる。誤った綴りは数多くの中から適当なものに訂正される。

現在の記述中に他のワードが残っていないときには、プログラムはステップ 9 7 0 からステップ 9 7 2 に進行して決定を下す。もし他のワードが残っているときには、プログラムはステップ 9 6 4 に戻り、次のワードの評価を開始する。このプロセスは与えられた記述中の夫々のワードに対して、および夫々の記述に対して、すべてのワードおよび記述が排出されるまで継続する。

Genic 3000 はこれまでのプロセスの部分として使用されるツールである。Genic 3000 は集積回路を含むカスタマーデータをデータ増大および変数指定するために使用される。Genic 3000 の出力は、後において前記プロセス拡張のためにノレッジベース 1 2 3 に入力され得る。

図 2 0 1 は Genic 3000 を使用してデータを処理するための代表的なデータ・フローを示している。Genic 3000 はアイテム 2162 中に示されるカスタマー・データ中に存在するヴェンダー・パート番号およびヴェンダー名を使用することによりデータ増大を完了する。ヴェンダー・パート番号およびヴェンダー名は、アイテム 2160 により描写されたヴェンダー・パートの発行されたデータベース中においてプログラム的に調べられる。データベース 2160 および 2162 はそれぞれステップ

2161 および 2163 において Genic 3000 によって読み込まれる。ステップ 2164 においては、発行されたデータベース中に見出された情報は、フォーマット 2165 中において ASCII テキストに変換され、ステップ 2166 においてノレッジベース 1 2 3 に取り込まれ得る。

発行されたヴェンダー・パート番号によりカスタマーデータ中に見出されたヴェンダー番号を照合させるプロセスは、直接的な照合でもよく、またいくつかの

発見を含むものでもよい。もしパート番号が直接的に照合できないときには、マニファクチャラーの名称またはコードが必要となり、またカスタマーデータからのヴェンダー・パート番号を翻訳しなければならない。上記の翻訳は、プレフィックス、サフィックスの削除およびベース・デバイス番号の引出しによって行なわれる。

照合アルゴリズムは、発行されたデータベースファイルを走査して、少なくともベース番号が照合する立候補部分を発行する。なお、マニファクチャラー名、部分番号プレフィックスおよび部分番号サフィックス等を記録する。ベース番号の異なった分類番号（すなわち、ベース番号により表された部分の種類）も決定される。これらの決定が完了した後、照合の質が決定される。

立候補した照合の質は表7に示される評価表に基づいて行なわれる。評価表は、表7に示されるKarnaughマップ(An Engineering Approach to Digital Design, William I. Fletcher, (1980), ページ134参照)縮小表である。この場合、Karnaughマップは、考えられるすべての問題の可能性ある組合わせを保証するために、およびマニファクチャラーの照合中の関係、分類番号、プレフィックス、およびサフィックスを容易に表し得るために使用される。夫々の表のセルは評価値を含んでいる。

行と列の索引は論理値であり、異なる位置が正しいか否かによって、表された値がその位置の値であるか否かに示している。これらの論理値は、関連した条件が正しいか否かを示しており、そしてそれらは順次に部分番号のこの位置においてなされた照合であるか否かを示している。マップの内容は、特別のセルの程度を示す完全な値である。このマップにおいては、下側のセル値(1)の方がより大なるセル値(12)より好ましい。

表7 サフィックス／マニファクチャラー

プレフィックス クラス	0 0	0 1	1 1	1 0
0 0	1 2	1 1	1 0	1 1
0 1	9	7	6	8
1 1	4	3	1	2
1 0	5	4	2	3

表7 照合グレード間の関係を検証するKarnaughマップ

表7中のKarnaughマップの表示を表8に示す。表8はグレードを照合するための照合条件から英語変換した表である。任意の与えられた照合のグレードは、コンポーネント照合の認識により決定され得る。ベース番号部分は、任意の次の照合テストがなされる前において照合しなければならない。

表8

レーティング	マニファクチャラー 照合	サフィックス 照合	1クラス照合	プレフィックス 照合
1 2				
1 1	Yes			
1 1		Yes		
1 0	Yes	Yes		
9			Yes	
7	Yes		Yes	
8		Yes	Yes	
6	Yes	Yes	Yes	
5				Yes
4	Yes			Yes
3		Yes		Yes
2	Yes	Yes		Yes
4			Yes	Yes
3	Yes		Yes	Yes
2		Yes	Yes	Yes
1	Yes	Yes	Yes	Yes

この説明の次に、この表は更に第3の説明においてプログラムの縮小される。プログラムにおいて、各照合変数は数字として指定され、そしてその仲間

付加されると、予め限定された値の整列中に唯一のインデックスを準備する。予め限定された値はそれら自身のグレード値である。

もし数値的な値が割当てられると、コンポーネント照合値は、下記のように各コンポーネントの照合のためのグレードを表示するために使用される。

表 9

記 憶	値
マニファクチャラー	1
サフィックス	2
1 クラス	4
プレフィックス	8

これらの照合値の組合わせは、0～15までの範囲によりインデックスを形成する。グレーディング整列の内容は、グレードインデックスを実際のグレード値に変換する表を調べるところの表10に示される。

表 10

インデックス	グレード値
0	12
1	11
2	11
3	10
4	9
5	7
6	8
7	6
8	5
9	4
10	3
11	2
12	4
13	3
14	2
15	1

下記は、ヴェンダー（マニファクチャラー）およびヴェンダー（マニファクチャラー）部分番号の使用により、本システムを通じて働く例である。

Intel 2901Bのヴェンダー部分番号を照合するための立候補ヴェンダー部分番号のリストは次のようになる。

立候補番号#1：マニファクチャラー＝AMD，部分番号＝LM2901B

立候補番号# 2 : マニファクチャラー=Intel, 部分番号=2 9 0 1 A
発行されたデータベース中において、マイクロプロセッサにより 1 分類中においてベース番号2901のみが発見されたと仮定する。

立候補番号# 1 は図 8 7 に示される例に翻訳される。

前述の照合軌範を使用して、この部分はサフィックスに照合し、および 1 クラス中に発見された。表 7, 8 または 1 0 から、この部分照合は 8 としてグレードされ得る。

同様にして、立候補番号# 2 は図 1 3 1 に示される例に翻訳される。

再び前述の照合軌範を使用して、この部分はマニファクチャラー、プレフィックスに照合し、および 1 クラス中に発見された。この場合において、両記述においては何等のプレフィックス情報も欠如することにより、プレフィックスは照合した。表 7, 8 または 1 0 から、この部分照合は 3 としてグレードされ得る。

低いレーティングの方が良いことから、立候補番号# 2 は「Intel 2901B」のために照合として選定される。

出力は、ノレッジベースに属する移入マップおよび移入ファイル中に挿入される発行されたデータベースからの変数情報を含む。

このソフトウェアの操作は図 1 6 5 ~ 1 6 7 のフロー図中に記述されている。操作は商用データベースが読込まれると開始する。このデータベースの内容が読込まれて、図 1 6 5 のステップ 9 0 0 においてベース番号が付けられる。データベースが読込まれた後に、ステップ 9 0 1 において、プログラムは入力部分データの読込みを開始する。ステップ 9 0 2 において、ステップ 9 0 1 により受け取られた部分番号がそのコンポーネント部分、すなわちベース番号、プレフィックス、サフィックスおよびマニファクチャラーに分解される。

ステップ 9 0 3 において、ステップ 9 0 2 中に見出されたベース番号がステッ

プ 9 0 0 のデータ中に照合データベース登録を発見するために使用される。アイテム 9 0 4 は照合すべき他のベース番号が商業データベースのデータ中に存在するか否かを決定する。もし照合するベース番号がないときには、プログラムはステップ 9 0 1 において、入力すべき他のヴェンダー名およびヴェンダー部分番号

を調査するため継続する。もし照合すべき1若しくはそれ以上のベース番号が発見されると、プログラムはステップ905に継続する。ステップ905において、プログラムはステップ904中に発見された各々の照合登録のために進行し、各照合アイテムのためのループに通じる単一のパスにより各照合アイテムを進行させる。ステップ905において、ステップ903中に発見された登録の1つが検索され、そのアイテムがステップ902において一致された部分番号のプレフィックス部分との照合を含むか否かを決定する。ステップ905において照合が発見されると、プログラムは図166のステップ906において、プレフィックス照合を示すフラグをセットするように進行する。プレフィックス照合の値は表8中に示されている。もしプレフィックス照合が発見されないときは、ステップ906はスキップされ、プログラムはステップ907に継続する。アイテム907は、この繰り返しにおいて、テストされるべき照合アイテムを検索する。テストされるべき照合アイテムは、ステップ903において発見されたものであり、ステップ902においても発見されたサフィックス部分との照合のためである。もしサフィックス照合が発見されると、操作はステップ908において継続し、サフィックス照合フラグをセットする。サフィックス照合フラグの値は、表8中に示されている。もしサフィックス照合が発見されないときには、プログラム操作はステップ909に継続する。

ステップ909においては、プログラムはこの繰り返しにおいてテストされるべき部分、すなわちステップ903において発見された部分を検索し、ステップ902において発見されたマニフアクチャラー部分との照合が行なわれる。もしマニフアクチャラー照合が発見されたときには、プログラム操作はステップ910に継続し、マニフアクチャラー照合を示す照合フラグがセットされる。マニフアクチャラー照合の値は表8に示される。もしマニフアクチャラー照合が発見されないときには、ステップ910はスキップされ、プログラムはス

テップ911に継続する。ステップ911においては、プログラムは、ステップ902において発見されたベース番号が照合し、かつステップ904において実際に照合したすべての部分が同一種類の部分であるか否か、または、換言すれば

、ステップ 904 において発見されたすべての部分が同一の機能を発揮するか否かの決定を試みる。もしそれらが同一であると決定されると、プログラムは図 167 のステップ 912 に継続する。ステップ 912 において、プログラムは、1 クラスの部分がステップ 901 において求められる部分番号のためのものであることを発見したことを示すフラッグをセットする。1 クラスフラッグの値は表 8 に示されている。もしステップ 904 において照合したセット部分が部分の複数クラスを表示すると決定されるか、その部分が異なる機能を行なうものであると決定されたときには、プログラムはステップ 913 に継続する。

ステップ 913 において、プログラムはステップ 906、908、910 および 912 における照合フラグセットを、表 9 に示されるグレーディング表中のインデックスを構成する。このインデックスは、ステップ 914 において、表 9 の内部整列表示中のグレードを調べるために使用される。ステップ 914 において発見されたグレードは、ステップ 914 において照合した部分を割り当てられ、このグレードは、ステップ 914 で開始されたこの繰り返しのために使用される現在の部分である。ステップ 915 においては、プログラムは、ステップ 914 において発見された、またこのプロセスによりテストされる必要があるところの更に他の照合があるか否かを決定する。もし他のアイテムが残っているときには、ステップ 903 において発見された照合リスト中の次のアイテムと共にステップ 905 に継続する。もし調べるべき残存アイテムがときには、プログラムはステップ 916 に継続する。ステップ 916 においては、プログラムは、ステップ 914 により最高のものが獲得される登録のためにステップ 905 において開始され、かつステップ 915 において終了される繰り返しプロセスの結果を調べる。次に最高のグレードはステップ 916 により選定され、最高の照合としてユーザに提供される。最高の照合は、前記登録により連想されるすべてのデータを包含する。前記登録により連想されるデータは、ステップ 900 において読み込まれたデータベースから得られる。

前記プロセスを容易に進めるためには、ノレッジベースを自動的に形成すること、および、多数の情報を部分的に独特に定義づけし、かつ可能な限り近接した

クラス別けしておく必要がある。一連の取込の効用を介して顧客データをノレッジベース中に取込むことにより目的が達成させられる。

一連の取込の効用は、多くの方法によりノレッジベースを変更することがある。最も明白な方法は、要求を創造すべき新規データのノレッジベースへの取込みによることである。取込の効用は、現実の要求によるデータ（変数）を付加し、または改修することがある。

取込み効用にはまた多くのものがある。すなわち、シーマの削除、実例の削除、他の属性へのテキストの変換、および既知の好ましい値および数値に対する未知の変数値の明記、ならびにクラスマップおよび顧客データからの情報に基づく目的クラスの大幅な変更等である。

現実には5つの取込の効用がある。すなわち、取込み、クラスマップ取込み、単純な取込み、取り込みA、および取込みBである。これらは作用的には同様であるが、それぞれのものは、ユーザデータにより連想される種々のシナリオを解決するのに必要な独特な特徴を持っている。

単純な取込み、クラスマップ取込みおよび取込みBは、新規の例を創造することによりノレッジベース中に顧客サイド情報を取込む。

取込みおよび取込みAは、部分情報を実在する選択された実例中に取込み、それらを増大させることを可能にする。

クラスマップ取込みは、クラスマップの使用により目標クラスおよびユーザデータ中の選択された分野を大幅に変更する。

取込み効用への第一の事物は取込みファイルである。取込みファイルは主に部分をクラス分けするために最も有効である顧客サイドデータからなる。取込みファイルは、取込み効用のために望ましい方法でフォーマットされていなければならない。取込みファイルは3つのセクションを有する。すなわち、クラスパスセクション、属性名セクションおよび顧客データセクションである。

クラスパスセクションは、取込みファイル中の第1のセクションである。それはタブで分離された名称からなる単一ラインのデータである。名称は取込み目的

クラスへのルートからパスを指定するクラス名称である。取込みは一般的に目的

クラスにおいて行なわれる。

属性名セクションは取込みファイル中の第2のセクションである。それはタブで分離された属性名からなる単一ラインのデータである。属性名セクションは顧客データが取込まれる属性を指定する。このセクション内で指定される属性名はクラスパスセクションを指定する目的クラスにおいて確実に根拠のあるものでなければならない。

顧客データセクションは取込みファイル中の第3のセクションである。それはタブで分離された値からなる1個またはそれ以上のラインである。属性セクションにはそれぞれ名づけられた属性のための値がある。属性セクションには行間における1対1の対応および顧客データセクションには顧客データ中のデータ列がある。

次の図は取込みファイルのフォーマットを示している。第1行はクラスパスセクションである。第2行は属性名セクションである。第3行および第4行はユーザデータセクションである。取込みが行なわれると、2つの実施が創出される。部分番号および記述属性のための変数が第1の実例に対して1 2 3 3 2 1および1/4 × 1 1/2 2 0 UNFとセットされ、かつ第2の実例に対して1 2 3 3 2 2および1/4 × 1 3/4 2 0 UNFとセットされる。

root Mechanical Part Number	Components Description	Fasteners	Bolts
123321	1/4 × 1 1/2 20UNF		
123322	1/4 × 1 3/4 20UNF		

現在の実例を変更することを許容するために、属性セクション内においてある属性が例えば「キー」属性として指定される。キー属性はある実例を検索しかつ選定するために使用される。それらの変数が実例に照合しなければ、取込みファイル内の「キー」属性列中の値は取込みの間において操作される。ストリング型の属性のみがキー属性として指定される。1個またはそれ以上の属性がキー属性

として選定される。キー属性は属性名セクション内のどの場所でも現れる。属性名は取込みが行なわれる際にキー名としてまた属性として使用できる。キー属性

は取込みファイルの属性セクション内において名称を接頭させることにより“key>”（例えばkey>Part no）と指定される。次の図はキー属性を含む取込みファイルの属性セクションを示している。

key >Part Number	Description	key >Vendor Code	Ven
Descript			

数属性の取込みを許容するために、属性セクション内においてデフォルトユニットを指定しなければならない。デフォルトユニットは、属性に連合するユニット群からのユニット名である。デフォルトユニット指定は属性名の次に符号“1”を介して付けられる（例えばLength | Inches）。次の図はユニット指定を伴う数属性を含む取込みファイルの属性セクションを示している。

Part Number	Description	Length Inches	Diameter Feet
-------------	-------------	-----------------	-----------------

属性値は取込みファイル内の顧客データの他のデータ列を付加することなく他の属性に等しを作られる。これは2個の属性名を符号“!”によって分離する指定によって行なわれる（例えば Description! Description 2）。次の図は等しを作られた属性を含む取込みファイルの属性セクションを示している。

Part Number	Description !	Description2
-------------	---------------	--------------

コメントは符号“#”を伴う行を開始することにより取込みファイル中に挿入することができる。空白行もまた取込みファイル中に設けることができる。

インポートの間に4つのフェーズが実行される。初期化、コマンド・ラインオプションの説明、ログイン、及びデータベースのオープンの後第1のインポートフェーズが開始する。

第1のフェーズは図153のステップ1300のインポートファイルのノン・コメントラインを読むことである。このラインはクラスバスセクションである。このクラスバスは、読み出されて、クラスネーム中へ区分される。クラスバスは

、デスティネーションクラスへのクラスパスに従って有効化される。デスティネーションクラスが存在すると、第2のフェーズが開始する。

第2のフェーズはインポートファイル（1302）の第2のノン・コメントラインを読み出す。これは、アトリビュートネームセクションである。このラインが読み出されるとアトリビュートネームが区分される。アトリビュートネームは、デスティネーションクラスに存在することが照合されることにより有効化される。単純なインポート及びインポートAに対して、すべてのアトリビュートはストリングの型になる。インポート、クラスマップインポート及びインポートB、すべての特定の数値のアトリビュートもまた有効なデフォルトユニット（特別なシンボル“|”）を明確にする。更に、アトリビュート名は、“key>”及び“!”のような特定の他のシンボルの要素に分けられる。

第3のフェーズは存在するインスタンス中にインポートが行われる時にだけ発生する。このフェーズの間、ルックアップテーブルが作成される（1307）。ルックアップテーブルはデスティネーションクラスにおける各インスタンス及びキーアトリビュート（例えば、アトリビュートネームセクション内の“key>”が前置されたアトリビュート）のためのパラメータ値に対しインスタンスハンドルを含む。ルックアップテーブルはインポートファイルのカスタマデータセクション内のデータにより増大されるインスタンスを高速に位置付けるために使用される。

第4のフェーズはインポートファイルのカスタマデータセクションからデータを読み出して、インポート（1308）を実行する。これは各ラインを読み取って、フィールド内にそのラインを区分して、その後適当なアトリビュートに対するパラメータを設定して値をインポートすることにより実行される。

デスティネーションクラスが自動的に選択される場合、カスタマデータから選択されたフィールドはクラスマップファイルからの一つのクラスと照合する試みに使用される（1327）。もし、デスティネーションクラスが識別できると、新しいインスタンスがそのクラス内に作成される（1328）（1323）。

新しいインスタンスが作成されている時には、そのインスタンスはパラメータの設定より前に作成される（1323）。

データが存在するインスタンス中にインポートされる場合、そのキー値は最初はそのデータから引き出され、その後バイナリの探索がルックアップテーブル上で全て存在するインスタンスにマッチすることを検出するために実行される（1324）。全てマッチするインスタンスが一旦見つかると、パラメータが残りのデータから設定される。

列挙されたアトリビュートが図154のステップ1311にインポートされる場合、もしインポートファイルからのアトリビュートデータがアトリビュートに関連して存在するどのイニュマレータ（enumerator）ともマッチしないなら、データをスキーマへの新たなイニュマレータに自動的に加えられるか、またはインポートユーティリティが存在するイニュマレータのメニューをそこから選択するために提供する（1317）。もし、メニューが提供されると、ユーザはファイルから読み出したデータを存在するイニュマレータへマップするか、そのデータを新たなイニュマレータに加えるために使用するか、またはデータを無視してパラメータを未定義のまま残すかの選択をする。ユーザが存在するイニュマレータにデータをマップする選択を行うと、この情報はインポートユーティリティにより保持され、その後の同じデータについて急に起きる事態に使用され、その時そのユーティリティは自動的に存在するイニュマレータへそのデータをマップする。

数のアトリビュートがインポートされる場合（1312）、インポートファイルからのアトリビュートデータが単純な数の文字であると（1318）、そのパラメータは、インポートファイルのアトリビュートネームセクション内で指定されたアトリビュートのためのデフォルトユニットを使用してセットされる。もし

、インポートファイルからのアトリビュートデータが数値または非数値の両方のデータを含んでいると、そのデータに含まれたユニットスペシフィアと見なされ、アトリビュートネームセクション内のに記載されたデフォルトユニットに取って代わるために使用される。そのユーティリティはそのデータをユニットスベ

シファイアの位置に区分けし、そのスペシファイアネームをアンノウンユニットネームとする（1319）。そうでない場合は、そのインポートユーティリティは存在するユニットネームを選択するためのメニューを提供する（1320）。メニューが提供された時、ユーザはそのファイルから読み出されたデータを存在するユニットネームにマップするか、そのデータを無視してパラメータを無定義のままに残すかの選択を行う。もしユーザが、そのデータを存在するユニットネームにマップする選択を行うと、この情報はインポートユーティリティにより保持され、その後の同じデータについて急に起きる事態に使用され、その時そのユーティリティは自動的に存在するイニシエータへそのデータをマップする。

ブーリアンアトリビュートがインポートされた場合（1310）、インポートファイルからのアトリビュートデータがTrue, Yes, T, Y, または1の中の一つであると、TRUE（真）の値と見なす。もし、インポートファイルからのアトリビュートデータがTRUEまたはFALSE（偽）の何れとも認識できないデータを含んでいると（1314）、そのインポートユーティリティは選択を行うためのメニューを提供する（1316）。そのメニューが提供されると、ユーザはファイルから読み出したデータをTRUEの値(value)にマップするか、そのデータをFALSEの値にマップするか、またはそのデータを無視してパラメータを無定義のままに残すかの選択を行う。もし、ユーザがそのデータをTRUEまたはFALSEの値にマップする選択を行うと、この情報はインポートユーティリティにより保持され、その後の同じデータについて急に起きる事態に使用され、その時にはユーティリティはそのデータを自動的に適切なブーリアンバリエーションにマップする。

次のセクションは、5つのインポートユーティリティのそれぞれについて、如何にそれらの特徴がこれらのユーティリティの一般的な記述と外れているか述べている。

インポートユーティリティの使用法及びシンタックスが図125に示されている。

る。

異なる操作が設定されたオプションに依存して実行することができる。もし、
-r オプションが設定されると、マッチングインスタンスがインポートされるよ

り知識（ノウリッジ）から削除されることになる。もし、-M オプションが設定されると、マッチングインスタンスが無いことが検出された場合、新たなインスタンスがキーアトリビュートフィールドを含むそのライン上の全てのデータから作成される。もし、-U オプションが使用されると、一つのマッチングインスタンスが存在した場合だけ、情報がインポートされる。1つのインスタンスより多くがマッチングすると何もインポートされない。もし、-X オプションが使用されると、マッチングインスタンスが無い時だけ情報がインポートされる。これは、-M オプションと共に使用されなければならない、そうでないと何の影響も与えない。

クラスマップインポートはノリージベースに新たなインスタンスを付加する。インスタンスは、そのクラスのデータ内でマッチするパターンを記述するマップに基礎を置く特定のクラスに付加される。

このコマンドの使用法は図 1 2 6 に示される。

初期化の後インポートマップファイルが読み出される。インポートマップファイルの第 1 のフィールドはそのデータ内でマッチするパターンであり、第 2 はインスタンスがインポートされる相手である CADIS-PMX クラスである。-O オプションにより指定される例外ファイルが作成される。例外ファイルは如何なるマップパターンともマッチしないパラメータであるためにインポートすることができないインスタンスを中に持っている。

インポートファイルのクラスパスセクション内のクラスパスは、アトリビュートネームセクション内でネーム付けられた全てのアトリビュートが有効であるクラスをネーム付けられなければならない。

カスタムデータがファイルから読み出される時、-f オプションで指定されたフィールドはマップファイル内でパターンをマッチするために使用される。もし、マッチが得られないと、そのインスタンスは -O オプションで指定されるインスタンス例外ファイルに出力される。

もし、-f オプションフィールドがクラスマップ内のパターンとマッチすると、そのインスタンスはそのクラスに付加され、そのパラメータがセットされる。

もし、パラメータのためのアトリビュートが列挙（イニユマレート）されたものであると、そのイニユマレートは存在しないスキーマに付加される。これは、インポートがDBXLockを取得することを意味する。

もし、インスタンスパラメータのためのリプレースメント（置き換え）のアトリビュートがブール数である場合、そのテキストが 'x', 'X', 'T', 't', 'TRUE 1', 'true', または '1' であるなら、パラメータはTRUEに設定される。

インポートマップのフォーマットは以下に示される。例えば, "1 0 inch spike" というバリューを持つインポートファイル内の第1のライン上の第1のデータフィールド上でインポートが実行されると仮定する。そのデータフィールドは最初にパターン "Thyristor" に対して比較される。このパターンはマッチしないので、そのデータフィールドは次に正規の表現のメタキャラクタを含むパターン "*spike*" に対して比較される。これはマッチするので、第1のラインは、ルート "PMX_Root" の下のクラス "Mechanical" の下のクラス "Spikes" にインポートされる。

```
Thyristor PMX __Root Electrical Discrete Thyristor
*spike* PMX __Root Mechanical Spikes
```

簡単なインポートは最も基本的なインポートユーティリティである。それは、インスタンスを作成するためだけに使用され、ストリングアトリビュートだけが指定される。コマンドの使用法は次のテーブル 1 1. に示される。

テーブル 1 1

```
simple _import[-u user][-p password]-d dbname[-p][-v]import_file
```

-u ログインを行っている時、ノンギブログイン i d が使用される
場合にユーザネーム 'username' を使用

-p ログインで指定されたパスワードを使用

-d 接続するための論理データベースネーム

-v バーボース(verbose) モードにターン

import__file インポート情報を含むファイルのネーム

No "key >" インポートファイルのアトリビュートネームセクション内
で許容されたアトリビュート。ストリング型のアトリビュ
ートだけがインポートされる。このユーティリティは、存
在インスタンスを更新するのに使用され、アトリビュート
のパラメータが現在定義されていない時だけである。

```
importA [-u user][-p password ] -d dbname [-p] [-v]
```

import__file

-u ログインを行っている時、ノンギブログイン i d が使用される
場合にユーザネーム 'username' を使用

-p ログインで指定されたパスワードを使用

-d 接続するための論理データベースネーム

-v バーボースモードにターン

import__file インポート情報を含むファイルのネーム

このユーティリティは新たなインスタンスを作成し、全てのアトリビュ
ート型のインポートを許容するために使用される。

```
importA [-u user][-p password ] -d dbname [-p][-v]-o nogoparts
import__file
```

-u ログインを行っている時、ノンギブログイン i d が使用される
場合にユーザネーム 'username' を使用


```

-p      ログインで指定されたパスワードを使用
-d      接続するための論理データベースネーム
-v      バーボースモードにターン
-o      パラメータ設定の困難さに起因するインポートすることができない
        部分を持つ部分データを書き込むためのファイルのネーム
import__file インポート情報を含むファイルのネーム

```

II. 付加的な構成及び変更

本発明はここでパートマネージメントの問題へのアプリケーションを参照しながら説明されてきたが、この技術について知識を有するものは、この開示による利益を受けた後、この発明が他の同様のアプリケーションに有用であると評価するであろう。例えば、本発明は機構が種々の記述を持つオブジェクトの多くのインスタンスの一つを確実に見つけた上でバリュウを置くような如何なるアプリケーションにも特に有用であろう。この中で記述されたダイナミッククラスマネージャは、クラス分けまたはスキーマを再構築することが望ましい如何なるアプリケーションにも特に有用である。

しかし、本発明はローカルエリアネットワークに関連付けて説明されたが、この技術で知識を有するものなら、この開示による恩恵を受けた後、他の構成及び実行が可能であると評価するであろう。例えば、このシステムは多数のユーザステーションを持つメイン・フレームまたは単体のコンピュータ上で実行することができる。このシステムは、またLANだけでなく、ワイドエリアネットワークまたはインターネットのようなネットワーク上で実行することができる。

付加的なファイルマネージャ140の展開は可能である。ファイルマネージャ140によりダイナミッククラスマネージャ134及びハンドルマネージャ137に供給されるインタフェースは、ダイナミッククラスマネージャスキーマ及び第2の持続性のストレージ103上のインスタンスデータのコピーを保持するために一致する。スキーマ及びインスタンスに変化が起きると、第2のストレージにも起きる。ダイナミッククラスマネージャ134はファイルマネージャ140を介して第2のストレージ103からデータを読み出すことにより初期化される

。他の第2のストレージ機構はインタフェースの仕様に従って実行することができる。他の実行は、リレーショナルデータベースマネジメントシステムを含む、Informixデータベース、Oracleデータベース、Raimaデータベース等の商業的なデータベースを使用することができる。他の実行は他の適当なファイルフォーマット使用して構築することができる。

A. 全般的なアーキテクチャー

本発明の現在の好ましい実施例が図204に示され、1またはより多くのノワリッジベースクライアント4112、4118及び4111及びノワリッジベースサーバ4108を包含するクライアント/サーバのアーキテクチャーを備えるネットワーク4100を採用する。図205に示す好ましい実施例において、ノワリッジベースサーバ4108は、オブジェクト指向のロックマネージャ4125、ダイナミッククラスマネージャ4134、コネクションマネージャ4135、クエリマネージャ4136、ハンドルマネージャ4137、ユニットマネージャ4138、データベースマネージャ4139、及びファイルマネージャ4140を包含する。サーバホスト4109は、予めローカルディスクドライブ4110上に格納されたソフトウェア及びノワリッジベース4123と共に、ノワリッジベースサーバ4108を動かすよう指名される。ノワリッジベースクライアント4131は、図示された実施例内のネットワーク4100を介してノワリッジサーバ4132と相互作用する。好ましいシステムでは、レジストリサーバ4141及びライセンスマネージャ4142がシステムへの非公認のアクセスを制御するために備えられている。レガシイクライアント4133及びレガシイマネージャ4145は、むしろオブジェクト指向データベースマネジメントシステムと連携して使用するため現在のレガシイデータベースの組織をスキーマ中に適合するために包含されている。アプリケーションプログラミングインタフェースまたはAPI4143は図示された実施例内にも示されている。

スキーマエディタ4144はスキーマまたはデータベース4123を修飾または変更するための備えられている。本発明による提供される同時制御により、多数のスキーマエディタ4144が同時に使用され、その間多数のリトリバ41

30が使用される。スキーマエディタ4144、ダイナミッククラスマネージャ4134、リトリバ4130、コネクションマネージャ4135、クエリマネージャ4136、ハンドルマネージャ4137、ユニットマネージャ4138、データベースマネージャ4139、ファイルマネージャ4140、レジストリサーバ4141、ライセンスマネージャ142、API143、レガシイマネージャ4145、及びノーリッジペースクライアント4131の構造と動作は上記においてより詳細に説明されている。

B. 同時制御

図204に図示された例において、多数のユーザまたはクライアント4111、4112及び4118がネットワーク4100が接続されて示されている。第1のクライアント4111は、ディスプレイ4116、マウス4117及びキーボード4122を備えるものとして示すサン・マイクロシステムのSPARCステーション4111上で動作する。第2のクライアント4112はディスプレイ4113、マウス4114及びキーボード4115を備えるものとして示すIBM互換コンピュータ4112上で動作する。第3のXウィンドウズのクライアント4118はコンピュータ4118、ディスプレイ4119、マウス4120及びキーボード4122を備えるものとして図示されている。

このシステムは、相互のクライアント4131を介して接続された1またはより多くのユーザによりインタラクティブな編集をサポートする。例えば、ユーザはスキーマエディタ4144を使用してアトリビュートの付加及び削除によりスキーマの変更、スキーマに対し全てのセクションを付加し、スキーマの階層内にスキーマの全てのセクションを再配置し、及びインスタンスの修飾、付加及び削除を行うことができる。これらのインタラクティブな編集動作は他のユーザが同時にリトリバ130を使用してデータベース4123にアクセスしている間に行うことができる。これらの同時動作の管理は、多数のユーザが同時にデータベース4123を変更している同じ時間に、多数のユーザがデータベース4123にアクセスする機能を包含し、これは同時制御に起因する。

本発明では、オブジェクト指向ロックマネージャ4125が、ユーザに対し他

のユーザにより修飾が施されている間にそれらのビューを破壊することなく、クラスオブジェクトのクエリーとビューを許容する同時制御メカニズムを備える。これらの修飾には、クラス、アトリビュート、インスタンス、及びパラメータの付加、削除及び編集が含まれる。

好適な実施例では、ロックマネージャ 4125 はクラスマネージャ 4134 のサブシステムである。

本発明はクラスオブジェクトに基づくロックインヘリタンスを使用して同時制御システムの実行を最適化する。ロックマネージャ 4125 は、ロックのサブクラスインヘリタンス無しにクラス上に置かれたロックのメカニズムを実行させる。このメカニズムは、“クラスロック”と呼ばれる。ロックマネージャ 4125 はまた、ロックに対するインヘリタンスメカニズムを備える。このインヘリタンスメカニズムは“ツリーロック”と呼ばれる。あるクラスのツリーロッキングは、子孫のクラス上のクラスロックの設定を物理的に要求することなしに、インヘリタンスによるそのクラスの全ての子孫上の“ロック”内に結果をもたらす。

本発明はクラスレベルのロックのグラニュラリティ（細粒性）を使用することによりロックされる必要があるオブジェクトの番号を簡易化する。これは実行効率を良好にする。クラスロックのグラニュラリティまたはスコープはそれ自身がクラスであり、クラスにより定義されたアトリビュート、及びそのクラスに関連するインスタンスがある。図 206C は、本発明に従ってロックの細粒の階層を図に描いた図式ダイヤグラムである。本発明の有効性は、インスタンスが所属するクラスと独立してインスタンスがロックされることが許されないということである。これは、図 206A 及び図 206B に示す方策と対立する。本発明では、クラスは、個別（クラスロック）か、またはグループ内（ツリーロック）の何れかでロックされるが、インスタンスはそうにロックされない。同時性は、問題になっているインスタンスがそれ自身ロックされるかを決定することによることなく、むしろそれが所属するクラスにロックされるかを決定することにより制御される。複合オブジェクトは一つのクラスである。

本発明はオブジェクト指向データベース内で 4 つの型が採用されているが、3 つの型のロックモードだけを使用し、同時制御を実行することができる。本発明

において使用するロックモードの3つの型は：クラスシェアロック（“CSL”）、ツリーアップデイトロック（“TUL”）、及びツリーイクスクルーシブロック（“TXL”）である。使用されるロックモードの第4の型は、ツリーシェアロック（“TSL”）であり、これは事実上はクラスシェアロックのグループであると考えられる。従って、好適な実施例では、ノーリッジベースサーバ4132は実際に4つのロック型：排他、アップデイト、及びシェアロックの2つの態様をサポートする。

“クラスシェアロック”は、また“CSL”として参照され、分配(sharing)のため単一クラスのノードをロックする。

“ツリーシェアロック”は、また“TSL”として参照され、分配のためそのクラスに根を下ろしたサブツリーをロックする。このロックは、ちょうどそのサブツリー内の各クラス上にCSLを設定するように振るまう。

“ツリーアップデイトロック”は、また“TUL”として参照され、インスタンスの編集のクラスに根を下ろしたサブツリーをロックする。これは、時々単に‘アップデイトロック’またはU-ロックと呼ばれる。

“ツリーイクスクルーシブロック”はまた、“TXL”または時には単にX-ロックとして参照され、排他的使用のためそのクラスに根を下ろしたサブツリーをロックする。

ノーリッジベース4123を変更するいくつかの動作は、ライト(write)ロックの排他的な型を要求することなく実行することができる。ライトロックの他の型は、ここでは“update”ロックとして参照され、バリエー、アディング、及びムービングインスタンスのパラメータの修飾を含む一定の動作に対して使用される。アップデイトロックはシェア及びイクスクルーシブロックの混合である。少なくとも1つのアプリケーションによりロックされたオブジェクトはアップデイトすることができるが、同時に1またはそれ以上のアプリケーションによりロックされたオブジェクトはシェアすることができる。これは、アップデイトロックを伴う1つのアプリケーションは他のアプリケーションによる試験が行われている同じ時間にそのオブジェクトを変更することができることを意味する。オブジェクトがアップデイトとシェアでロックされている時に発生することができるノ

ーリッジベースへのこれらの変更は、アプリケーションの認識及び管理を極めて容易にすると考えられる。

アップデートロックはイクスクルーシブロックではなくライトロックの“Weak”型である。ノーリッジベース 4 1 2 3 に対するいかなる変化もライトロックが要求され獲得されることを必要とする。アップデートの動作のいくつかは、イクスクルーシブロックを必要とし、他のアップデートの動作はアップデートロックを必要とする。しかし、アップデートロックを要求とするアップデートの動作の一つは、“at least” アップデートロックを必要とする。イクスクルーシブロックはノーリッジベース 1 2 3 へ変更を行うのに常に充分であるが、アップデートロックは選択された変更のセットを作成するのにより好都合で、より同時的なロックである。

ノーリッジベースクライアント 4 1 3 1 は、これらのメカニズムを使用するツールに最大の有用性、安定性、及び実効性を与える適切な細粒及び継承のロックを設定するため、ロックマネージャ 4 1 2 5 により提供されたオブジェクト指向ロックメカニズムを使用する。ここに記載した例は、リード指向データベースシステムに最適化できる。これは、パーツ管理に使用されるノーリッジベーススキーマ内で特に有効である。

ロックは2つの目的に役立つ。第1は、ロックはアプリケーションまたはノーリッジベースクライアント 4 1 3 1 により使用され、あるオブジェクトが試験されていることを知らせたりステートメントを作成する。これは、同じオブジェクトの試験を同時に行う多重のアプリケーションに対し害を与えないからであり、この目的に使用されるロックの型はシェアロックである。これに対し害を与えないからである。いくつかのアプリケーションは、同時的なシェアロックにより、オブジェクトをシェアすることができる。典型的には、アプリケーションは、シェアロックを、スキーマを介した案内、クエリの実効、及びインスタンスの試験を行うものとして使用する。

アプリケーションによるロックの第2の利用は、オブジェクトを変更したいことを知らせることである。そのアプリケーションは、同じオブジェクトを変更しようとする他のアプリケーションが存在しないことが保証される。このロックの

型はライトロックと呼ばれる。他のアプリケーションはライトロックされたオブジェクトを変更するのを妨げられる。典型的には、インスタンスの付加または削除、パラメータバリュウの修飾、またはスキーマの編集の時にアプリケーションはライトロックを使用する。上記で指摘したように、ノーリジベース差4132はライトロックの2つの型：イクスクルーシブロック及びアップデットロックをサポートする。イクスクルーシブロックは、問題の原因となる方法でアプリケーションが相互作用するのを防ぐために使用される。例えば、インスタンスを削除すべき時、またはスキーマが編集される時、イクスクルーシブロックが使用される。あるオブジェクトが問題の原因とならない方法で変更することができる場合、むしろウイカー（weaker）アップデットロックが最大の同時性を防ぐために使用される。

本発明で使用されるロックの大多数は‘ツリー’ロックであると判断されるであろう。上記に議論において、オブジェクト（実際にはクラス）をロックすることについて言及した。実際に重要なことは、クラスはロックの影響下にあるということである。与えられたクラスの先祖のクラスが排他的（イクスクルーシブ）にロックされると、そのクラスもまた、排他的にロックされたクラスにより根を下ろしたサブツリー内にあるため、有効に排他的にロックされる。

アプリケーションは要求によりロックを確立する。もし、要求が成功すると、そのアプリケーションはロックを獲得する。そのアプリケーションは、そのオブジェクトをそれ以上ロックされる必要がなくなるとそのロックを解除しなければならない。その要求は、他のアプリケーションにより既に要求された他のロックとのロックの衝突に負けるであろう。その衝突はライトロックを要求して、そのオブジェクトが既にライトロックされている場合、または要求がシェアロックであって、そのオブジェクトが既に排他的にロックされている場合に起きる。

ロックできるオブジェクトは常にクラスである。インスタンスは決してロックされない。

この好適なシステムは、サブツリーをインスタンスのための別名として使用する。この方法で、より少ないロックが適用されて、複雑さが少なく高速なシステムとなる。クラスでないいくつかのオブジェクトを変更するアプリケーションに対

し、そのオブジェクトに関連するクラスにライトロックが要求される。これを言い変えると、インスタンスにライトロックを加えるにはインスタンスが加えられるべきそのクラスに対し要求されなければならない。パラメータバリューは、そのアプリケーションがインスタンス内に所有するクラス上にライトロックを要求する時にだけ変更することができる。例えば、スキーマデベロッパまたはエディタ 4 1 4 4 は、あるクラスにより定義されたアトリビュートに対し変更を行うため、そのクラスにイクスクルージブロックを要求する。

ロックマネージャ 4 1 2 5 及び ノーリッジベースサーバ 4 1 3 2 はアプリケーションがロックを要求することができる前にロックホルダとなることを要求する。それは `pmx_スタートロックホルダ()` 機能を使用することにより、ロックホルダになり、こうしてロックホルダを開始する。`pmx_スタートロックホルダ()` 機能はソフトウェア機能セクションにおいてより十分に説明される。ノーリッジベースサーバ 4 1 3 2 及びロックホルダへのアプリケーションの接続の結合は、ロック間の衝突を解決するためにアプリケーションを他のアプリケーションとどのようにに区別するかにある。アプリケーションは多重ロックホルダを開始することができ、こうしてアプリケーション内のロック要求の衝突を引き起こす。これは、分離される必要があるアプリケーション内のサブルーチンにとって有用である。アプリケーションはロックホルダを終了することによりアプリケーションがロックホルダであることを停止する。

ノーリッジベースサーバへの各アプリケーションの接続は、図 2 0 5 内に示す唯一のロックホルダテーブル 4 1 4 6 を備える。ロックホルダテーブル 4 1 4 6 は各接続に対する現在のロックホルダを管理するロックマネージャ 4 1 2 5 により使用される。

図 2 5 5 はロックホルダテーブル 4 1 4 6 のデータ構造を示す。この実施例では、ロックホルダテーブル 4 1 4 6 はブーリアンバリューのダイナミックリストである。

ロックホルダテーブル 4 1 4 6 内の TRUE (真) バリューは、開始されたロックホルダを表す。ロックホルダテーブル 4 1 4 6 内の FALSE (偽) バリューは終了したかまたは一度も使用されなかったロックホルダである。ロックホルダテー

ル4 1 4 6へのインデックスはロックハンドル2 6 7それ自身である。こうして、図2 5 5内に示す例において、TRUEバリュー4 6 0 1はロックホルダハンドルゼロであり、開始されている。ロックホルダハンドル4 2 6 7は参照数字4 6 0 2により一致が識別されたテーブル4 1 4 6の人力に相当し、それは開始されていたことを示すTRUEバリューを持っている。バリューFALSE 4 6 0 3を持つロックホルダハンドル2は終了されている。

ロックホルダの開始動作は図2 5 6内のフローチャートに示されている。ステップ4 6 0 7において、ロックホルダテーブル4 1 4 6は、使用されてなく割り当てができるロックホルダを表すFALSEのバリューにより検索される。もし、FALSEのエLEMENTが検出されると、そのテーブルインデックスは、“newLH”に指定される。ステップ4 6 0 8において、もし、FALSEエLEMENTが見つかったと制御はステップ4 6 0 9に進む、そこではインデックス“newLH”のロックホルダテーブル4 1 4 6エLEMENTが、ロックホルダが割り当てられたことを示すためにTRUEにセットされる。もし、ステップ4 6 0 8においてFALSEエLEMENTが見つからないと、制御はステップ4 6 1 1を継続し、そこでロックホルダテーブル4 1 4 6の終端に新たなエLEMENT4 6 0 6が割り当てられ、この新たなエLEMENT4 6 0 6のインデックスが“newLH”に指定される。制御はステップ4 6 0 9を継続する。ステップ4 6 1 0では、インデックス“newLH”が新たに開始したロックホルダハンドルとして復帰する。

図2 5 7はロックホルダを終了する動作のフローチャートである。そのプロセスは本発明により一つのステップ4 6 1 2内で非常に速く実効される。ステップ4 6 1 2において、ロックホルダハンドルにより終了されるものとして索引付けられたロックホルダテーブル4 1 4 6エLEMENTはFALSEにセットされる。

図2 0 8は本発明で提供されたロック型及び細粒によるロック衝突を表す説明図である。最初のコラム4 2 2 0は最初のユーザにより保持されたロックを表し、それはロックホルダ1として参照される。先頭の列4 2 1 9は第2のユーザにより要求されたロックを表し、それはロックホルダ2として参照される。衝突状態は交差するセル内に示される。そのセルはロックホルダ2によって要求されたロックがロックホルダ1によって保持されたロックと衝突しているかどうかを示

す。例えば、もし、参照番号4216により示すコラム4220内の位置により表されるようにロックホルダ1がTULまたはクラスを持つなら、参照番号4217により示された列4219内の位置により表されるようにロックホルダ2がCSLを要求し、その時は交差するセル4221はロックの衝突が無いことを示し、ロックホルダ2はクラス上にGSLを得る。テーブル12は本発明により使用できる有用なロックの型のリストであり、ロックの細粒及びそれらの記憶である。最も制限されたロックのメカニズムはたった一つのロックホルダだけが認められるイクスルーシブロックである。最も寛大なロックの型は衝突の無い型である多重ロックホルダを許容するシェアロックである。同時性についての中間レベルはアップデートロックにより用意される。オブジェクト指向のロックマネージャはクラスイクスルーシブロックまたはクラスアップデートロックを備えが、本発明の好適な実施例に使用されるロックの型に対するツリー細粒は、眺め(view)の安定性を用意するのに十分である。シェアロックはクラス及びツリー細粒の両方に備えられているのが望ましいが、それは本発明に要求されたものではない。

好適な実施例において、同時制御は主としてアプリケーションレベルで起き、DBM(データベースマネージメント)レベルでは起きない。クライアントアプリケーション4130、4144またはAPI4143の4133はアプリケーションが機能を試みる時明確にロックを要求しなければならない。この記述では、時々ユーザまたはロックを“要求”するロックホルダを参照する、好適な実施例では、GUIプログラムがこの動作をユーザから隠し、ユーザが実際にクライアントアプリケーション4130、4144またはAPI4143の4133がロックを要求していることを気がつかないという意味で、そのような要求を明確に実行することを必要としないように、GUIプログラムが書かれている。クライアントアプリケーションは、ユーザが階層または編集パーツの操縦を試みる時、例えばレトリバー4130またはスキーマエディタ4144を使用する時、ロックマネージャ4125に対し背景の要求を遂行する。もし、衝突が探索され、または要求が失敗すると、そのユーザは適当なダイアログボックスまたは、ユーザが操縦または編集を試みたスキーマ部分へのアクセスが実現できなかったというメッセージを介して知らされる。好適なシステムにおいて、クライアントアプリケー

ジョン4130、4144及び4133は同時制御を達成するために良く振る舞われ、協力している。言い換えれば、同時性はアプリケーション4130、4133及び4144の協力により調停される。

与えられたアプリケーション及びロックホルダの結合は、衝突無しで同じクラスのための同じ型の多重ロックを要求することができる。例えば、図208を参照する上記の記述において、同じユーザはロックホルダ4001とロックホルダ4002になることができる。これは、例えば、同じユーザが第2のウィンドウをオープンする時に起きる。アプリケーションにより獲得した各型のロックの勘定はノーリッジベースサーバ4132のロックマネージャ4125により維持される。複数のロックはそれが要求されたのと同じ回数開放されなければならない。しかし、好適な実施例では、ロックは5つの方法でひとまとめに開放することができる。ノーリッジベースサーバは多重ロックを開放するための2つのAPI機能をサポートする。ロックホルダにより獲得された全てのロックはロックホルダが終了する時間開放される。そして、アプリケーションにより獲得された全てのロックはアプリケーションがノーリッジベース4123を閉じる時またはアプリケーションがログアウトする時間開放される。

ノーリッジベースサーバ4132のロックマネージャ4125によりサポートされたシェアロックは忠告的である。これは、シェアロックは他のアプリケーション（インスタンスまたはスキーマの編集を希望したアプリケーション）に対し、スキーマの一部が操縦されたことを通知するという意味である。シェアロックは、スキーマの操縦のため、またはインスタンスのクエリまたは試験のために要求されないが、それらはむしろ選択される。シェアロックの獲得は他のアプリケーションが強制されたライトロックの獲得を防止する。ロックマネージャ4125は及びノーリッジベースサーバ4132は如何なるスキーマまたはインスタンスが適切なライトロック無しに編集されるのを許容されない。従って、もしレトリバ4130、スキーマエディタ4144、レガシー4133、またはユーザが書いたAPIプログラムのような、API 4143のクライアントが、それらの一つをスキーマの一部内に操縦する時は何時でもシェアロックを要求し、それは操縦されている間スキーマに起きるかもしれない如何なる変化からも絶縁される

。

クライアントアプリケーション4130、4144及びAPI 4143の4133は、クラスデスクリプタまたはそのクラスののためのアトリビュートデスクリプタを取得する時は常にクラスののためのクラスシェアロックを要求すべきである。この方法はデスクリプタ内のデータが有効であり、有効として残ることを保証する。クライアントアプリケーション4130、4144及び4133は、またクエリを行うクラスにおいてクラスツリーロックを使用すべきである。これは、例えば、クエリが適用されるサブツリー内でインスタンスを削除するような、他のアプリケーションを妨げるために使用される。

本発明の中にはロックは包含されない。オブジェクトは同じ型の多重のロックを持つ。ロック要求及び開放は対になっている。図示された実施例において、クラスシェアロックの開放を実行する機能は、オブジェクト上の一つのクラスシェアロックだけを開放することになる。

ロックマネージャ4125の動作は図209-211を参照することにより良く理解される。図209はオブジェクト指向データベースの一部の例を表すクラス階層4215の図式的なダイアグラムである。この例では、クラス4202は記述された全ての他のクラスに対する先祖である。もし図209が完全なデータベースを描写したなら、クラス4202はルートクラスになるであろう。クラス4202はクラス4201及びクラス4205の親である。クラス4201は、クラス4206及びクラス4200として示すように2つの子を持つ。クラス4205はクラス4210及び4207の親である。クラス4200は2つの下位を持ち：クラス4203とクラス4204である。クラス4206はクラス4208とクラス4209として示すように2つの子を持つ。同様に、クラス4210とクラス4207はそれぞれ2つの子：クラス4211と4212、及びクラス4213と4214のそれぞれ、と共に示される。

もし、ロックがクラス4200に対し要求されると、第1のステップは要求されたロックがこのクラス4200における他の如何なるロックと衝突するかチェックする。これは図209に表され、クラス4200が、階層4215内のこのポイントにおいて衝突するロックのためのクラス4200を試験するステップを

表す黒い正方形として示されている。衝突の決定は図208内に表されてマトリクスに従って行われる。もし、クラス4200に対する要求されたロックがクラスシェアロックCSLであり、クライアント4200が既にクラスシェアロックCSL、ツリーシェアロックTSLまたはツリーアップデートロックTULに従っている場合には、何の衝突も存在しないで、回答“NO”（すなわち、衝突無し）が返される。これは図208において、CSL、TSL、及びTUL列とCSLのカラムの交差位置に“NO”として表されている。もし、クラス4200に対して要求されたロックがクラスシェアロックCSLであり、クラス4200が既にツリーイクスクルーシブロックTXLに従っていると、その時は衝突があり、回答“YES”（すなわち、はい、衝突有り）が返される。これは図5において、CSLカラムがTXL列と交差する位置に“YES”として表されている。もし、衝突があると、要求されたロックは許可されない。

それから、ロック要求手順はこの特定の例において、要求されたロックがクラス4200の先祖の4201及び4202で他のどのロックと衝突するか否かをチェックするステップに継続する。これは図210に表され、そこにはクラス4201と4202が階層4215内のこれらのポイントで衝突するロックについて先祖のクラス4201及び4202を試験するステップを表す黒い方形として示される。衝突の決定は図208内に表されたマトリクスに従って実行される。クラス4200は図210内においてロックが要求された先のクラスであるクラス4200を示す陰影がつけられた方形として表されている。図6においてチェックがうまく完了した後、先祖のクラス4201及び4202は衝突についてチェックされる。この例では、クラス4200上のロックの要求はクラスまたはツリーロックの何れかになることができる。もし、衝突が示されると、その要求されたロックは許可されない。もし、衝突無しが検出されると、回答“NO”が返される。このような場合、要求されたロックがクラスシェアロックであるなら、その要求されたロックは許可される。もし、要求されロックがツリーイクスクルーシブロック、ツリーシェアロックまたはツリーアップデートロックであるなら、その手順は図211に関連して記述されたステップに続く。

図211は、図209及び図210内でのチェックが良好なら、クラス420

0上のロック要求を許可するプロセスにおける後続のステップの間の階層を図示するダイアグラムである。降下クラス4203及び4204は衝突についてチェックされる。クラス4203及びクラス4204はそれぞれ、階層4215内のそれらのポイントで衝突するロックについて降下クラス4203と4204を試験するためのステップを表す黒い方形として示される。衝突の決定は図208内で表されたマトリクスに従って実行される。クラス4200は図211内で、クラス4200がロックの要求された先のクラスであることを示すため陰影がつけられた方形として表されている。もし、衝突が示されると、回答“YES”が返されて、要求されたロックは許可されない。もし、衝突無しが検出されると、回答“NO”が返されて要求されたロックは許可される。

ロックマネージャ4125の動作は図247-254を参照することで十分理解される。動作の期間、ロックマネージャ4125は図254に示されるダイナミックロックテーブル4400を維持する。ロックテーブル4400はスキーマと相互作用する。例えば、もしあるクラスがスキーマから物理的に付加または削除されると、ロックテーブル4400はそれに従って変更される。スキーマにより反映された受け継いだパターンに基づいたシステムによりロックは評価される。ロックテーブル4400は図示された例中のノーリッジベースサーバ4132により維持されている。

図254内に示されたロックテーブル4400は好適な実施例において、各列がスキーマ内のクラスに対応するよう組織されている。各カラム（行）はシステムが使用するロックホルダに対応する。ロックテーブル4400の各制御は以下の記述で参照するために番号が付けられている。例えば、クラスハンドル4003に対応する列とロックホルダ4002に対応する行の交差部は参照羽4410により示される。もし、クラスシェアロックがロックホルダ4003に対応するユーザによりクラスハンドル4005に対応するクラス上に置かれると、ロックマネージャ4125はロックテーブル4400のエLEMENT 4419内に示すCSLを置くであろう。この技術の知識を有するものなら本発明によれば同時制御システム内にインスタンスロックするための用意が無いことが理解される；ロックテーブル4400だけがクラスのために用意されている。

もしロックホルダ4006がクラスハンドル4004に対応するクラス上にロックの幾つかの型を置く試みをする、ロックマネージャ4125はロックホルダ4001がそのクラス上に衝突ロックを持つかを決定するためロックテーブル4400のエレメント4404を必ずチェックする。ロックホルダ4006がクラス上に置こうと試みたロックの型と衝突するロックの型が何であるかの決定は図208のロック衝突テーブルに従って決定される。もし、衝突するロックがセル4404で見つけ出されないと、その場合はロックマネージャ4125は、どのロックホルダ4002がクラスハンドル4004に対応するクラス上に衝突のロックを持つかどうかの決定をするためセル4411のチェックに進む。もし、見つからないと、ロックマネージャ4125はロックホルダ4003がクラス上に衝突のロックを持つかどうかを決定するためセル4418のチェックに進む。ロックマネージャ4125は残りのロックホルダ4004、4005及び4007に対応する全てのセル4425、4432及び4446がそれぞれチェックされるまで継続する。これは基本的に図209に表されたプロセスに対応する手順である。

たとえば図210に示されたチェックングクラス4201のような、先祖のクラスのチェックを行うため、ロックマネージャ4125はロックマネージャ4125にどのクラスハンドルがクラス4201に対応するかについての情報を供給するためのメカニズムを持たなければならない。ダイナミッククラスマネージャ4134がこの機能を実行する。こうして、図210に描かれた先祖のチェックを実現するため、ダイナミッククラスマネージャ4134はロックマネージャ4125に先祖クラス4201のためのクラスハンドルを供給する。もし、対応するクラスハンドルがクラスハンドル4002であるなら、ロックマネージャ4125はクラスハンドル4002に対応する列内のセル4402、4409、4416、4423、4430、及び4444について、クラスハンドル4004の列を参照した上記した方法で、チェックを実行することができる。

同様に、図211に示された下降クラス4203及び4204についてチェックを実行するため、ダイナミッククラスマネージャ4134はロックマネージャにこれらのクラスに対応するクラスハンドルを供給し、ロックマネージャはロ

テーブル 4 4 0 0 の対応する列を衝突するロックがあるかを決定するためチェックする。動作にインスタンスを含む時は、ダイナミッククラスマネージャ 4 1 3 4 はロックマネージャ 4 1 2 5 にそのインスタンスのためのクラスを供給し、システムはそのクラスと衝突するロックについてチェックする。

ロックが要求された時、ロックマネージャはロックの衝突を識別するためコネクションとロックホルダハンドル 4 2 6 7 の両方を使用する。スキーマまたはインスタンスの編集が試みられる時は、ダイナミッククラスマネージャ 4 1 3 4 は、最初に動作をロックマネージャ 4 1 2 5 から実行するための許可を求める。一つの実施例では、許可のためのチェックにコネクションだけが使用される。この例では、適切なロックの存在についてチェックする時に編集動作を求められたロックホルダは考慮されない。この効果は、この特定の例において、各API編集機能への入力の前兆としてロックホルダハンドルを要求するのを防止することによりなされる。

図 2 4 7 はスキーマの編集を行うための許可を要求するためのステップを描いたフローチャートである。イクスクリューシブロックは望まれたスキーマの編集を行うためにロックホルダにより要求される。ステップ 4 4 5 0 にいて、カレントクラスがチェックされるべきクラスと同じにセットされる。ステップ 4 4 5 1 において、“カレントクラス”はそれが排他的にロックされたか、（すなわち、それがツリーイクスクリューシブロックTXLを持つか）を見るためにチェックされる。図 2 5 4 を参照すると、もし要求するロックホルダがロックホルダ 4 0 0 3 であり、カレントクラスがクラスハンドル 4 0 0 3 であると、このステップは事実上イクスクリューシブロックのための交差セル 4 4 1 7 をチェックする。もし、それが排他的にロックされていると、その時はこの例では編集を行うことが試みられているロックホルダにより排他的にロックされていることを意味する。その場合、ロックマネージャ 4 1 2 5 はステップ 4 4 2 5 において要求を行うロックホルダ 4 0 0 3 に対応するクライアント 4 1 3 1 に対し“OK”の表示を返す。もし、それが排他的にロックされていると、フローはステップ 4 4 5 2 に進み、そこでロックマネージャ 4 1 2 5 は“カレントクラス”がルートクラスであるかどうかの決定をするためにチェックする。もし、それがルートクラスであると、ロッ

クマネージャ 4 1 2 5 はステップ 4 4 5 4 において “no” を返す。そうでない場合、フローはステップ 4 4 5 5 に進み、そこで “カレントクラス” は “カレントクラス” が存在したクラスの親クラスと同じにセットされる。ロックマネージャ 4 1 2 5 はダイナミッククラスマネージャ 4 1 3 4 に親がどれかを探ね、その情報はクラスマネージャ 4 1 3 4 によりロックマネージャ 4 1 2 5 に供給される。それからこの手順は図 2 4 7 に示されるようにステップ 4 4 5 1 にループバックする。事実上、ロックマネージャ 4 1 2 5 はこの手順を使用して先祖のチェックを行う。図 2 4 8 はインスタンスの編集をするための許可を要求に対するステップを描いたフローチャートである。インスタンスの編集を実行するため、イクスクルーシブロックまたはアップデートロックが要求される。ロックマネージャ 4 1 2 5 は最初にクラスマネージャ 4 1 3 4 に対し、そのインスタンスが依存するクラスがどれであるかをロックマネージャ 4 1 2 5 に告げるようクラスマネージャ 4 1 3 4 に尋ねなければならない、そしてこの情報はダイナミッククラスマネージャ 4 1 3 4 により提供される。ステップ 4 4 5 7 では、彼クラスはチェックされるクラスと同じにセットされる。ステップ 4 4 5 8 では、“カレントクラス” がイクスクルーシブでロックされているかアップデートでロックされているかを見るためにチェックされる（すなわち、それがツリーイクスクルーシブロック TXL またはツリーアップデートロック TUL をもつかどうか）。もし、イクスクルーシブかアップデートかでロックされていると、この例では編集を行っているロックホルダによりそのようにロックされたことを意味する。その場合、ロックマネージャ 4 1 2 5 はステップ 4 4 5 9 において要求を行うロックホルダに対応するクライアント 4 1 3 1 に対し “OK” の表示を返す。もし、それがイクスクルーシブでなくロックされていると、フローはステップ 4 4 6 0 に進み、そこでロックマネージャ 4 1 2 5 は “カレントクラス” がルートクラスであるかどうかを決定するためのチェックを行う。それがルートクラスであると、ロックマネージャ 4 1 2 5 はステップ 4 4 6 1 において “no” を返す。そうでないと、フローはステップ 4 4 6 2 に進み、そこでは “カレントクラス” が “カレントクラス” の存在したクラスの親クラスと同じにセットされる。ロックマネージャ 4 1 2 5 はダイナミッククラスマネージャ 4 1 3 4 に、どれが親かを探ね、その情報は

クラスマネージャ 4 1 3 4 によりロックマネージャ 4 1 2 5 に供給される。その後の手順は、図 2 4 8 に示すようにステップ 4 4 5 8 にループバックする。

図 2 4 9 はクラスシェアロックを要求するステップを描いたフローチャートである。ステップ 4 4 6 4 において、“カレントクラス”がロックが要求された先であるクラスと同じにセットされる。ステップ 4 4 6 5 において、そのクラスは何かの他のロックホルダにより排他的にロックされたかを決定するためチェックされる。もし、そうであるなら、ロックマネージャ 4 1 2 5 はステップ 4 4 6 6 において“no”を返す。もしそうでないなら、ロックマネージャ 4 1 2 5 はステップ 4 4 6 7 に進み、そこでロックマネージャ 4 1 2 5 は“カレントクラス”がルートクラスであるか否かを決定するためチェックを行う。もし、そうであるなら、ロックマネージャは“yes”を返し、ステップ 4 4 6 8 において要求された CSL を許可する。もし、そうでないなら、ロックマネージャ 4 1 2 5 はステップ 4 4 6 9 に進み、そこでロックマネージャ 4 1 2 5 はクラスマネージャにどれが親クラスであるか尋ねる。その情報がロックマネージャ 4 1 2 5 に供給される時、“カレントクラス”は親クラスと同じにセットされ、フローはステップ 4 4 6 5 にループバックする。

図 2 5 0 はツリーシェアロックを要求するためのステップを描いたフローチャートである。ステップ 4 4 7 0 で、“カレントクラス”はツリーロックが要求されたクラスと同じにセットされる。ステップ 4 4 7 1 で、“カレントクラス”はそれが何かの他のロックホルダにより排他的にロックされたか否かを決定するためにチェックされる。これは、要求を行うロックホルダに対応するカラム（行）内のセルを除く全てのセルで“カレントクラス”に対応するロックテーブル 4 4 0 0 内の列をチェックする。もし、そうであるなら、ロックマネージャ 4 1 2 5 はステップ 4 4 7 2 において“no”を返す。もし、そうでないなら、ロックマネージャ 4 1 2 5 はステップ 4 4 7 3 に進み、そこでロックマネージャ 4 1 2 5 は“カレントクラス”がルートクラスであるか否かを決定するためのチェックを行う。もし、そうでないなら、ロックマネージャ 4 1 2 5 はステップ 4 4 7 4 に進み、そこでロックマネージャ 4 1 2 5 は“カレントクラス”を親クラスと同じにセットする（ロックマネージャ 4 1 2 5 はクラスマネージャ 4 1 3 4 から親ク

ラスの確認を得なければならない)。その後、手順はステップ4471にループバックする。これは祖先のチェックにおいて有効な結果となる。もし、ステップ4473においてそれがルートクラスであることが検出されると、ステップ4475においてロックマネージャ4125は降下クラスの全てがチェックされたかどうかを見るためにチェックする。チェックされていると、その時はロックマネージャ4125は“yes”を返し、ステップ4476において要求されたTSLを許可する。そうでないなら、ステップ4477においてロックマネージャ4125は“カレントクラス”を未だに試験されたことがないある降下と同じにセットする。

次にステップ4478でロック・マネージャ4125は、新しい「カレント・クラス」が他のロック・ホールドによって排他ロックされているかどうかを決定するためのチェックを行なう。これによって實際上、要求ロック・ホールドに対応する列（column）の中のセルを除くすべてのセルでロック・テーブル4400内の対応する行（row）をチェックすることになる。新「カレント・クラス」が何か他のロック・ホールドによって排他ロックされていない場合には、フローはステップ4475へとループバックする。このループは實際上、子孫（descendants）をチェックする結果となる。新「カレント・クラス」が他の何らかのロック・ホールドによって排他ロックされている場合には、ロック・マネージャ4125はステップ4479で「ノー」を返す。

図251はトリー更新ロック（tree update lock, TUL）を要求するための諸ステップを示すフローチャートである。ステップ4480で「カレント・クラス」は、そこでこのトリーロックが要求されるもののクラスとイコールに設定される。ステップ4481で「カレント・クラス」は、何か他のロック・ホールドによって排他ロックまたは更新ロックされているかどうかを決定するためのチェックを受ける。これは、「カレント・クラス」に対応するロック・テーブル4400内の行を、要求ロック・ホールドに対応する列のセルを除くすべてのセルでチェックするものである。それがロックされていなければ、ロック・マネージャ4125はステップ4482で「ノー」を返す。ロックされていなければ、ロック・マネージャ4125はステップ4483に進み、そこでロック・マネージャ4125は「カレント・クラス」がルート

・クラス (root class) かどうかを決定するためのチェックを行なう。それがルート・クラスでなければ、ロック・マネージャ4125はステップ4484へと進み、そこでロック・マネージャ4125は「カレント・クラス」を親クラスとイコールに設定する (ロック・マネージャ4125はこの親クラスの識別をクラス・マネージャ4134から受け取らなければならない)。次いで、手順はステップ4481へとループバックする。これは実際上、先祖 (ancestors) をチェックする結果となる。ステップ4483でそれがルート・クラスであると判明すれば、ロック・マネージャ4125はすべての子孫クラスがチェック済みかどうかを確認するためのチェックをステップ4485で行なう。それらがチェック済みならば、ロック・マネージャ4125はステップ4486で「イエス」を返し、被要求TULを認める。そうでなければ、ロック・マネージャ4125はステップ4487で、「カレント・クラス」を末検査の何らかの子孫とイコールに設定する。

次にロック・マネージャ4125はステップ4488で、新「カレント・クラス」が何か他のロック・ホルダによって排他ロックまたは更新ロックされているかどうかを決定するためのチェックを行なう。これは実際上、新「カレント・クラス」に関するロック・テーブル4400内の対応する行を、要求ロック・ホルダに対応する列の中のセルを除くすべてのセルでチェックする結果となる。新「カレント・クラス」が何か他のロック・ホルダによって排他ロックされていないならば、フローはステップ4485へとループバックする。このループは実際上、すべての子孫をチェックする結果となる。新「カレント・クラス」が何か他のロック・ホルダによって排他ロックまたは更新ロックされていれば、ロック・マネージャ4125はステップ4489で「ノー」を返す。

図252はトリー排他ロック (tree exclusive lock, TXL) を要求するための諸ステップを示すフローチャートである。ステップ4490で、「カレント・クラス」はそこでこのトリーロックが要求されるところのクラスとイコールに設定される。ステップ4491で、「カレント・クラス」は何か他のロック・ホルダからロックがかかっているかどうかを決定するためのチェックを受ける。これは「カレント・クラス」に対応するロック・テーブル4400内の行を、要求ロック・ホルダに対応する列の中のセルを除くすべてのセルでチェックする。何か他のロック

・ホールダが「カレント・クラス」上に何らかのタイプのロックをかけていれば、ロック・マネージャ4125はステップ4492で「ノー」を返す。そうでなければ、ロック・マネージャ4125はステップ4493へと進み、そこでロック・マネージャ4125は「カレント・クラス」がルート・クラスかどうかを決定するためのチェックを行なう。それがルート・クラスでないならば、ロック・マネージャ4125はステップ4494へと進み、そこでロック・マネージャ4125は「カレント・クラス」を親クラスとイコールに設定する（ロック・マネージャ4125は親クラスの識別をクラス・マネージャ4134から受け取らなければならない）。次いで手順はステップ4495へと進み、そこでロック・マネージャ4125は新「カレント・クラス」が何か他のロック・ホールダからTSL (tree shared lock, ツリー共有ロック)、TULまたはTXLがかかっているかどうかを決定するためのチェックを行なう。ロックがかかっているならば、フロアはステップ4496で「ノー」を返す。ロックがかかっていなければ、フロアはステップ4493へとループバックする。ステップ4493で「カレント・クラス」がルート・クラスであると判明すれば、ロック・マネージャ4125はステップ4497で、すべての子孫クラスがチェック済みとなっているかどうかを確認するためのチェックを行なう。すべてがチェック済みであれば、ロック・マネージャ4125はステップ4498で「イエス」を返し、被要求TXLを認める。そうでなければロック・マネージャ4125は、ステップ4499で「カレント・クラス」を、まだ検査を受けていない何らかの子孫とイコールに設定する。

次にロック・マネージャ4125はステップ4500で、新「カレント・クラス」が他の何らかのロック・ホールダから何らかのタイプのロックを受けているかどうかを決定するためのチェックを行なう。これは實際上、新「カレント・クラス」に関するロック・テーブル4400内の対応する行を、要求ロック・ホールダに対応する列の中のセルを除くすべてのセルでチェックする結果となる。新「カレント・クラス」が他の何らかのロック・ホールダからいかなるタイプのロックも受けていなければ、フロアはステップ4497にループバックする。このループは實際上、すべての子孫をチェックする結果となる。新「カレント・クラス」が他の何らかのロック・ホールダから何らかのタイプのロックを受けていれば、ロック・マネージャ4125はステップ4501で「ノー」を返す。

クライアント4131がリトリバ4130を起動すると同時実行制御システム (currency system) は図212に示した手順を実行してリトリバ・ウィンドウ4290を開く。図212は、リトリバ・ウィンドウ4290が開かれたときに実行されるロッキングプロセスを表すフロー図である。ステップ4225で、ユーザはリトリバ・ウィンドウ4290を開こうと試みる。ステップ4226では新ロック・ホールダが要求される。新ロック・ホールダの要求がステップ4226で失敗して、フローがステップ4227へと進めばクライアント4131はリトリバ・ウィンドウを表示しない。新ロック・ホールダ要求が認められれば、フローはステップ4228へと進む。

新ロック・ホールダはそのユーザと対応付けられる。ユーザとロック・ホールダの間には1対1の対応が存在する場合も少なくない。しかし、単一ユーザが複数のロック・ホールダとなりうる場合もあるので、以下の説明ではロック・ホールダの名称を使用する。図212に示した手順において、新ロック・ホールダは次にルート・クラスのためのCSL (class shared lock、クラス共有ロック) をステップ4228で要求する。図示例では、クライアント4131に対応付けられたGUIがルート・クラスのためのCSLを要求する。被要求CSLが認められなければ、フローはステップ4227へと進み、リトリバ・ウィンドウは開かれないうだろう。ステップ4227では、システムによってユーザ向けメッセージが生成されるようにするのが好ましい。ステップ4228で被要求CSLが認められれば、フローはステップ4229へと進み、リトリバ・ウィンドウがそのロック・ホールダのために開かれ、ユーザのディスプレイ4116に表示される。

図213は、クラス階層内のあるクラスが選択されたときにシステムによって実行されるプロセス4230を示す。ユーザがステップ4232でクラスの選択を試みると、ステップ4233において、そのユーザの知識ベース・クライアント4131に対応付けられたGUIによってCSL要求が出される。この要求が失敗すればフローはステップ4234へと進み、そのクラスは選択されない。CSLが認められれば、フロー・メソッド (flow method) はステップ4235に進み、そこでそのクラスは被選択クラスとなり、強調表示されるようになり、また対応付けられた属性が表示される。図216は、ユーザがクラストリー4248をナビゲートする際にユーザのディスプレイ4116上に表示される可能性のある画面の一例を示す。ルート・クラス4245

はディスプレイ内ではクラス4001と命名される。クラス4240はルート・クラス4245の子孫であり、ディスプレイ内ではクラス4002と命名される。クラス4241もまたルート・クラス4245の子孫であり、ディスプレイ内ではクラス4003と命名される。さらに、クラス4247もルート・クラス4245の子孫であり、ディスプレイ内ではクラス4006と命名される。クラス4241には2つの子孫、すなわちクラス4246とクラス4243がある。クラス4246は図216に示したディスプレイ内ではクラス4004と命名される。またクラス4243はディスプレイ内ではクラス4005と命名される。図示例では、ユーザはクラス4243を選択している。図213に示したメソッド4230で、CSLが認められた場合には、クラス4243は被選択クラスとなり、強調表示の4244となり、対応付けられた属性4242が表示される。

図214は、サブクラスを調べるためにクラスを開くプロセス4231のフロー図である。図216を引き合いにすれば、この例ではユーザが、開かれるべきクラス4241上でダブルクリックすると、図214のステップ4236でCSL要求が出される。CSLが認められれば、メソッドはステップ4237へと進み、クラス4241の表示が（クラス4240について図216に示したような）閉じたフォルダから開いたフォルダ4241に変わり、すべてのサブクラス（4246および4243）が表示される。開いたクラス4241のためにCSLを得るステップ4237は、図214に示したメソッドでは単一ステップとして図示されているが、ステップ4237は図213に示すステップ4233、4234および4235と同様に複数ステップで構成されることは言うまでもない。

図215は、ユーザがステップ4238で「ファインド・クラス (find class)」アクティビティを選択したときに生じるプロセスの諸ステップを表すフロー図である。（ファインド・クラス・アクティビティは、クラス階層またはスキーマを対象としたクラス探索である。）探索パターンに合致するクラスが、図213に示したプロセス4230を用いて最初を選択される。図213に示したプロセス4230が首尾よくいけば、図214に示したプロセス4231を用いてクラスが開かれる。図215のステップ4230および4231が図213および214にそれぞれ示した複数ステップ手順に対応することは、当業者には自明であろう。

ロック・マネージャ4125はスキーマ内の各クラスについて、また各ロック・ホー

ルダについて、ロック・テーブルを維持する。これは図217～219を参照すれば

もつとよく理解できよう。

図217は、図216のディスプレイに対応するスキーマ4248の線図であり、スキーマ4248内のクラス4245、4240、4241、4246、4243および4247の対応する内部ロック状態を示す。図218はロック・マネージャ4125によって維持されるロック・テーブル4250を示し、また図217に示されており図216に表示されているスキーマ4248に対応する。図219は、図218に示したロック・テーブル4250内のクラス4243に関するロック・オブジェクト4260のコンテンツを示す線図である。

図216に表示したスキーマ4248は図217に示すように線図で表せば、ロック・マネージャ4125によって維持されているクラス4245、4240、4241、4246、4243および4247の内部ロック状態を示すことができる。図214および図215で説明したクラスを開き選択するためのプロセスは、図217に示すスキーマ4248上ですでに実行されている。クラス4245とクラス4241はすでに開かれている。クラス4243はすでに選択されている。

ロック状態はロック・マネージャ4125によってロック・テーブル4250内に保存される。ロック・テーブル4250の参照数字4251、4252、4253、4254および4255によって識別される行はスキーマ4248内のクラス4245、4240、4241、4246、4243および4247にそれぞれ対応する。各ロック・ホルダは対応する列をもつ。図218に示すロック・オブジェクト4256、4257、4258および4259がそれである。これらのロック・テーブルの諸要素はスキーマ4248内のクラス4245、4240、4241、4246、4243および4247のクラス・ハンドル4251、4252、4253、4254および4255をロック・オブジェクト4256、4257、4258および4259に相関させる。ロック・テーブル4250内のクラス・ハンドル4251はロック・ホルダ4257に対応付けられたCSLロック・オブジェクト4261をもつ。なぜなら、スキーマ4248内のクラス4245が、ロック・ホルダ4257に当たるユーザのディスプレイ4116上で開かれているからである。スキーマ4248内のクラス4241は、ロック・ホルダ4257に当たるユーザがやはりそれを開いているため、CSL 4262をもつ。スキーマ4248内のクラス4243は、被選択クラスとなっているため、CSLロック・オブジェクト4260をもつ。もちろん

ん、このロック・ホルダ4257に関してクラス・ハンドル4254に対応するロック・オブジェクト4269は図218では空（empty）である。これは、図217に

示した対応するクラスがまったくロックを受けていないためである。同様に示した対応するクラス4249も図218ではブランク、すなわち空である。これも、図217に示した対応するクラス4240に、まったくロックがかかっていないためである。

被選択クラス4243に対応するロック・テーブル4250の要素4260の例を図219に示す。クラス4243に関するロック・オブジェクト4260のコンテンツには、対応するクラス4243に何らかのタイプのロックがかかっているかどうかを指示する手段が含まれている。図解した実施態様ではCSLカウンタ4263が、このクラス4243に関して1個のCSLが存在することを指示している。ロック・ホルダ・ハンドル4267は、ロック・マネージャ4125によって各ロック・ホルダを識別するために使用される。新ロック・ホルダ4226の要求が認められると（図212を参照）、その新ロック・ホルダに対してロック・ホルダ・ハンドル4267が割り当てられる。このように、新ロック・ホルダの要求を認める手順には、新ロック・ホルダにロック・ホルダ・ハンドルを割り当てるステップが含まれる。図解した実施態様では、各ユーザは一意のユーザ識別名、すなわちユーザIDをもつ。ロック・オブジェクト4260には、ロック・ホルダ・ハンドル4267に対応するユーザのユーザID 4268に関するレコードが含まれる。単一ユーザが多重ロック・ホルダとなる可能性もあるため、他のロック・ホルダ4256、4258または4259に関するユーザIDもロック・オブジェクト4260に関してはユーザID 4268と同じになる場合もある。

図219の例では、ロック・ホルダ4267はクラス4243にクラスロックをかけているが、いかなるタイプのトリロック（TSL、TULまたはTXL）もクラス4243に對しにかけていない。そのため、TXLロックのカウント4264はゼロである。同様に、TULロックのカウント4265およびTSLロックのカウント4266もこの例ではみなゼロである。

図220は、上述のユーザが図216～219に関して知識ベース4123内のクラス4243

にパートを追加したときに生じるプロセスの線図である。ユーザがデータベース4123にインスタンスを追加するためにステップ4270でリトリーブ4130を用いて「メイク・パート」機能を選択すると、クライアント4131はステップ4271で被選

択クラス4243に関してツリー更新ロック (TUL) を要求する。このTUL要求が認められれば、フローはステップ4273へと進み、ユーザはパートを追加するためのアクセスを認められる。次いで、アッド・パート (add part) オペレーションが完了すると、このTULはロック・マネージャ4125によって解除される。TUL要求が認められなければフローはステップ4272へと進み、ユーザはパートを作るためのアクセスを拒絶される。好ましい実施態様では、アクセスが不許可となった場合に、ステップ4272におけるかかるイベントを知らせるためのメッセージがユーザに送られる。

図222は、図220で説明したパート追加プロセスに対応するロック・テーブル4250の状態を示す。図221は、それに対してパートが追加される場所のスキーマ4248の線図である。パートが追加されるのは図221に示すスキーマ4248内のクラス4243である。図224は、これらの環境下でのパート追加プロセス中の画面表示である。アッド・パート機能を実行するにはTULが必要となる。それが認められれば、クラス4243に関するロック・オブジェクト4260はアッド・パート・オペレーションのためのTULをもつことになろうし、またスキーマ4248内のクラス4243が被選択クラスでもあるため、図222に示すようにCSLをもつことになろう。もちろん、このロック・ホルダ4257に関するクラス・ハンドル4254に対応するロック・オブジェクト4269は図222では空である、というのは図221に示した対応するクラス4246が何のロックも受けていないからである。同様にして、ロック・オブジェクト4249も図222ではブランク、すなわち空である。これもやはり、図221に示した対応するクラス4240が何のロックも受けていないためである。

図223はこの例におけるロック・オブジェクト4260を示す。この例では、このクラス4243に関するこのロック・ホルダにはすでにTULが認められているため、TULタイプに関するカウント4265は1である。前述のように、このロック・ホルダはクラス4243に関してCSLをも認められているため、この例ではCSLタイプ

に関するカウント4263も1である。図217～219および図221～223の同様の参照数字は同様の要素を示すので、図217～219に関しては説明を繰り返さない。

図220のステップ4273が好ましい実施態様で実行されるときには、「アッド・パート・ウィンドウ」4275（図224に示す）を開くステップも実行される。TULの影響下にあるトリー4276はアッド・パート・ウィンドウ4275では、クラス4245、クラス4241およびクラス4243を表す線図4276によって表される。

図225に関しては、ユーザがステップ4280でエディット・パーツ（edit parts）機能を選択すると、システムはステップ4281で、リトリーバ4130によるそのポイントまでのナビゲーションの結果として現在表示されている階層4248の対応する部分に関して既存CSLのコピーを作る。図227でエディット・パーツ機能は、スキーマ4248に対応するクラス・トリー4285のビュー（view）を含む新ウィンドウ4283を創出する。クラス階層トリー4285のその追加ビュー4283を提示するには、被提示クラス4245、4241および4243に関して新共有ロックを獲得しなければならない。これは、このウィンドウ4283内で表示または編集されることになるパートに関する一貫したビュー4285を保証するものである。システムはこのパーツ・エディタトリー4285に関する同一ナビゲーション・ロックを再発行することになる。

エディット・パーツ・ウィンドウ4283ではユーザは図228に示すスキーマ4248をナビゲートし（図226のステップ4282を参照）クラス階層トリー4285内のさまざまな場所に至ることができよう。このナビゲーションは、図226に示すように、前述の同じナビゲーションステップ4230および4231を用いる。

図229では、このユーザに関するロック・ホールダテーブル4280が、エディット・パーツ・ウィンドウ4283の創出が完了した後に示される。ロック・ホールダ4257には、クラス・ハンドル4255によって識別されるクラス4243に到達するためにすでに開かれた各クラス4245、4241および4243に関して2つのCSL（クラス共有ロック）4261、4262および4260が含まれている。すなわち、オリジナル・リトリーバ・ウィンドウ4290のために開かれた各クラス4245、4241および4243に関して1つのCSLがあり、またエディット・パーツ・ウィンドウ4283のために開かれ

た各クラス4245、4241および4243に関して1つのCSLがある。ユーザがトリー4285を下るナビゲーションを続けるとき、その際に通過する各クラスに関してCSLが獲得されることになろう。

図230はこの例に関してロック・オブジェクト4260をもっと詳しく示している。CSLカウントは2である。これは、図229に示した要素4260には、クラス・ハンドル4255に対応する行とロック・オブジェクト4257に対応する列の交差点で2つのCSLが含まれるからである。

図228～230の同様の参照数字は図217～219および図221～223の同様の要素を示すので、図217～219および図221～223に関しては説明を繰り返さない。

図231は、ユーザが被選択パート4330をあるクラス4243から与えられたサブトリー4248内の別のクラス4241に移動しようとする場合に用いられるメソッドのフローチャートである。図232は、任意の個数のパートをサブトリー4248内のあるクラス4243から同じサブトリー4248内の別のクラス4241に移動するという一般ケースで用いられるメソッドのフローチャートである。これら2つの図の違いは移動すべきパートの個数による。移動すべきパートが1個の特殊ケースでは、図231に示すメソッドの最適化をはかり、より小さな集合のコンポジット・オブジェクトにロックがかかるようにして、オペレーション完遂の見込みを強めることができる。

図232は、任意の個数のパートをサブトリー4248内のあるクラス4243から同じサブトリー4248内の別のクラス4241に移動するという一般ケースを示す。好ましい実施態様では、このメソッドは複数パートの移動に用いられ、また図231のメソッドは単一パートの移動に用いられる。図232では、エディット・パーツ・オペレーションの開始（図225）時に選択されたサブトリー4243に関してTXL（ツリー排他ロック）を獲得しようとするオペレーションが始まる。このロックが拒否されれば、オペレーションは拒絶される。このロックが認められれば、宛先クラス4241に関するTUL（ツリー更新ロック）が要求される。TULが認められれば、必要なロックがすべて保持され、出所クラス4243から宛先クラス4241へとパートが移動される。

図231はただ1つのパート330（4330の誤り?）が移動される特殊ケースである。この場合の唯一の違いは、TXLがどこで要求されるかである。前のケース（図232）でも機能することは判明しているが、その中でパートが移動されるとこ

ろのサブツリー4285に幅広いロックをかける必要があるため、成功の見込みが低くなろう。パート4330を移動しうる見込みを高めるために、移動対象のインスタンス4330を所有しているクラス4243に対してTXLがかけられる。これはツリー4285の可能なかぎり最小の部分にロックをかけることによって、最小個数のインスタンスだけがロックされるようにする。ロックが認められれば、オペレーションが図232に示した一般ケースの場合と同じ要領で進行する。

図231にもっと詳しく言及すれば、ユーザはステップ4300でプロセスを開始する。この最初のステップは「ユーザが1つのパートを移動する」と銘打っているが、もっと正確には、（必要なロックが認められた場合の）指示された機能のユーザによる実行の試みと理解されるべきである。同時実行制御システム（concurrency control system）は次にステップ4301へと進み、そこでこのシステムは、移動対象のパートを所有するクラス4243に関するTXLを要求する。図231は「パートの定義付けクラス」と呼ばれているが、「そのパートに関する所有クラス」と呼ばれるほうがもっと正確なことは当業者には理解されよう。TXLが（ロック・テーブル4250に矛盾するロックが存在することをロック・マネージャ4125に検知されるために）認められなければ、システムはステップ4302へと進む。好ましいのは、GUIがユーザに、被要求移動は実行できないことを、たとえばその情報が現在使用中であるためアクセスが拒否されている旨のメッセージをもって、通知することである。TXL要求がロック・マネージャ4125によって認められれば、システムはステップ4303へと進む。システムは宛先クラスに対するTULを要求する。TUL要求が（ロック・テーブル4250に矛盾するロックが存在することをロック・マネージャ4125に検知されるために）認められなければ、システムはステップ4304へと進むが、その際、要求された移動は実行できない旨をユーザに知らせるのが好ましい。TUL要求がロック・テーブル4250内の既存ロックと矛盾しない場合には、ロック・マネージャ4125は要求されたTULを認め、ステップ4305へと進む。そこで

、パートの移動が可能になる。

ダイナミック・クラス・マネージャ4134は、先にもっと詳しく解説した知識ベース4123内のオブジェクトに対するオペレーションを実行することになる。

図233は、図232で説明したパート移動の一般ケースに関するロック・テーブル4250を示している。ロック・ホルダのこのテーブル4250は、トリー4285の多数の部分に複数のロックをもっている可能性がある。図示したロック・テーブル4250は、このロック・ホルダ4257によって保持されるロックを識別する（リトリーバ4241によって保持されるロック、パーツ・エディタ4283によって保持されるロック、およびパーツ移動オペレーションのために保持されるロック）。クラス・ハンドル4251、4253および4255によってそれぞれ識別されるクラス4245、4241および4243はそれぞれ、リトリーバのためのCSL（クラス共有ロック）およびパーツ・エディタのためのCSLをもっている。さらに、クラス・ハンドル4253によって識別されるクラス4241は、クラス4241を宛先としてまさに移動されようとしているパートを追加するためのTUL（トリー更新ロック）をもっている。また、クラス・ハンドル4255によって識別されるクラス4243は、クラス4243からパートを移動するためのTXLをもっている。

図示した例では、クラス・ハンドル4255によって表されるクラス4243にTXLが認められるようにするために、現在、操作中の他のロック・ホルダ4256、4258または4259によって保持される他のTSL（ツリー・シェア・ロック）、TUL、TXLまたはCSLがなくても良い。このロック・ホルダ4257によって保持されるCSLがあるという事実そのものが、セルフ・コンフリクト状態とみなされる。TXLのリクエスト4257によってCSLロックが持されるという事実のために、この状態は許可され、TXLが認められる。一般にこのような状況では、識別可能なコンフリクトが要求を行っているロック・ホルダ4257とのコンフリクトである場合にのみ、幅広いロックが許可される。

選択パート4330移動のためのディスプレイの好適例を図235に示す。好適実施例では、図235に示したパーツ・エディタ・ウィンドウ4283は、波線の長方形429

1、ハイライティング（強調表示）、カラー・コードまたはその他の識別可能な特徴によってツリー4295(4285?)中のソース・クラス4243を視覚的に表示しなければならない。デスティネーション・クラス4241は、ハイライティング4292またはその他の識別可能な特徴によって視覚的に表示されなければならない。ユーザーは、移動コマンド・ボタン4335をクリックすることによって、移動機能を実行

することができる。

図236は、1つのパート4328が知識ベース4123から削除される場合の最適化ケースのフローチャートである。このプロセスは、ステップ4320によって開始される。ステップ4321では、削除するインスタンス4328を所有するクラス4243のためにTXLが要求される。図236では、クラス4243は「パートの定義クラス」と説明されているが、この分野に熟練した者であれば、クラス4243がより正確には所有クラスと呼ばれるものであることを理解できるであろう。TXLが得られない場合には、操作はステップ4322で拒絶される。操作が成功した場合には、ステップ4323でTXLが認められ、パート4328が削除される。

図238は、クラス4243からインスタンス4328を削除しようとするロック・ホルダ4257によって保持されていなければならないロック4260を示したものである。この状態は、パートをクラス4243から除去しなければならない場合の移動操作（図233参照）の一部と本質的に同じである。ロック条件は、それぞれクラス・ハンドル4251と4255によって表されるクラス4245および4243に関して同一である。クラス・ハンドル4253によって表されるクラス4241は、リトリバのためのCSLとパーツ・エディタのためのCSLを保持する。パート4328を削除するには、そこからインスタンス4328またはインスタンスの集合を除去するクラス4243のためにTXLを保持しなければならない。

図237は、ステップ4324から開始される、サブツリー4243から複数のパートを削除する場合の一般的なケースのフローチャートである。ステップ4325では、パーツ・エディタが呼び出された際に特定されるクラス4245からTXLが要求される。操作が実施されるサブツリー4285を定義するのは、クラス4245である。TXLの獲得に成功した後、所有クラスである4245、4240、4241、4246、4243または4247

からインスタンスが削除される。TXLが拒絶された場合には、操作が拒絶され、パートは削除されない。

図239は、パート削除操作に伴う好適ディスプレイを示したものである。パート4330は、ディスプレイ上のこのパートをクリックすることによって選択される。選択されたパート4330は、図238に図式的に示した、削除されるパート4328に対応する。図示した例では、削除パートは、選択したクラス4243によって定義さ

れるか、またはその一部である。選択したパート4330の削除は、図239に示した削除コマンド・ボタン4331をクリックすることによって開始される。好適実施例では、要求したロックが許可された場合には、システムは図240に示したダイアログ・ボックス、すなわちウィンドウ4332を開き、選択したパート4330の削除を希望するかどうかの確認をユーザーに求めてくる。削除操作は、“YES”ボタン4333をクリックすることによって確認される。このボタンをクリックすると、ダイアログ・ボックス4332が閉じられ、選択したパート4330が削除される。

図241は、スキーマの構造を変更するためにスキーマ・エディタを使用する際にコンカレンシー・コントロールが利用される場合のステップを説明したものである。ステップ4340でユーザーがスキーマ・デベロッパまたはスキーマ・エディタ4144を選択すると、ユーザーが変更を希望するサブツリー上でTXLロックを獲得する操作が次に行われる。この操作を完了するための手順は、アクティブなクラス4243のためにトリートリ他ロックを要求するステップ4341で開始される。TXLが獲得できない場合には、プロセスがステップ4342に分岐し、スキーマ・デベロッパ4144を始動できなくなる。TXLロックが認められた場合は、プロセスはステップ4343へと進み、スキーマ・デベロッパ・スクリーン4350が表示される。ステップ4343の後、スキーマ編集のために選択されたクラス4243上のリトリバ4290から獲得されたCSLロックがステップ4344で解除される（そのクラス4243のためにTXLロックが獲得されたため）。ステップ4345では、スキーマの編集が行われるクラス4243の親クラス4241のために、スキーマ・デベロッパ4144によってCSLロックが獲得される。CSLは、クラス4241の親クラス4245のためにもCSLロックを獲得する方が望ましい。

図242は、図241で説明した操作中に保持されるロックを表示するロック・テーブル4250を示したものである。スキーマ・エディタ4144によって、クラス・ハンドル4255によって表されるクラス4243のためにTXL(エレメント4260の中)が保持されることに注意されたい。これによって、システムを使用している他のユーザーはクラス・ハンドル4255によって表されるクラス4243以下のサブツリー上のいずれかの情報にアクセスするのが防止される。図242には、ロック・オブジェクト4260の詳細も示した。

図243は、図241のステップ4343で開かれるスキーマ・デベロッパ・ウィンドウ4350を示す好適実施例のためのスクリーン・ディスプレイを図示したものである。その中でスキーマを編集するクラス4243は、ハイライトされた状態3439で表示されることが望ましい。

図244は、インスタンスを表示する際にコンカレンシー・コントロール手段によって使用されるメカニズムを説明したフローチャートである。この操作は、ユーザーがパート・ディスプレイ・メカニズムを選択することによって、ステップ4360で開始される。図245では、この操作は、ユーザーがディスプレイ・コマンド・ボタン4352をクリックすると開始される。システムが要求された情報を表示するには、ロックが存在していなければならない。ロックを獲得するには、ソフトウェア4131がロック・ホルダにならなければならない。新たなロック・ホルダの要求は、図244で示したステップ3631で行われる。ロック・ホルダを求める要求が拒絶されると、プロセスはステップ4362に進み、ユーザーはパーツの表示を許可されない。しかし、ロック・ホルダ要求が認められると、プロセスはステップ4363に進み、ソフトウェア4131がユーザーのためにTSL(スリー・シェア・ロック)を要求する。TSLが拒絶されると、プロセスはステップ4362に進み、操作の継続が不可能になる。アクティブ・クラスに関してTSLが認められると、プロセスはステップ4364に進み、サブツリー4243の内部に含まれる情報が一貫している(coherent)という信頼をもってパートを表示することができる。

図245は、ロック・テーブル4250、スキーマ4248のダイアグラムおよびロック・オブジェクトのいずれかに関係する詳細を示したものである。図245は、図244

で説明した場合のロック・ホルダ・テーブル4250の状態を示したものである。リトリーバ4290は、クラス・ハンドル4255で表されるクラス4243に到着する前に巡航した、それぞれすべてのクラス4245、4241および4243のためのCSLロック4261、4262および4260を保持している。クラス4243で定義されたパートをシステムが表示するためには、新たなロック・ホルダ4258が形成され、クラス・ハンドル4255で特定されるクラス4243のためのTSLロック4373が要求される。TSL4373は、他のロックが許可されないようにし、それによって、スキーマ4248またはこのサブトリー4243内部のインスタンスの変更を不能にする。このように図246

に示した検索結果ウィンドウ4351に表示されるパート・リスト4365は一貫しており、ロック4373が保持される間、一貫性を維持する。

図245では、トリー・シェア・ロックは、クラス・ハンドル4255に相当する行とロック・ホルダ4258に相当する列との交差点4374のロック・テーブル4250によって示されている。また図245では、ロック・オブジェクト4373がより詳細に示されている。ロック・オブジェクト4373のTSLカウントは、クラス・ハンドル4255に対応するクラス4243のTSLロックがロック・ホルダ4258によって保持されているため、ひとつものとして示されている。ロック・ホルダ・ハンドル4372が2つなのは、これが図238から242で説明した上記のロック・ホルダとは別のロック・ホルダであることを示している。しかし、ユーザーID4370は4100として示した。これは、（ユーザーIDが4100の）同じユーザーが2つのロックを保持しているからである。

図253は、知識ベース・クライアント4131によるロック・マネージャ4125の適用を示すチャートである。リトリーバ・ウィンドウ4502が開かれると、コンカレンシー・システムがユーザーにロック・ホルダ4001(LH1)のポジションを割り当て、そのロック・ホルダにCSLを許可する。”ファインド・クラス”機能4503を実行するために、ロック・ホルダ4001は、図253に示したように、CSLを要求する。パーツ・エディット・ウィンドウ4504が開かれると、ロック・ホルダ4001はカレント・クラス上でTSLロックを獲得しなければならず、CSLはその階層を巡航する。スキーマの編集には、トリー・ロック(TULとTLX)が要求される。

パート・ディスプレイ・ウィンドウ4505を開くためには、ユーザーは新たなロック・ホルダ(LH2)の割当を受けなければならない、TSLを要求しなければならない。スキーマ編集ウィンドウ4506を開くために、図253が示しているように、LH1はTXLを要求する。パート追加または作成ウィンドウ4507を開くために、LH1はTULを要求する。好適システムでは、ユーザーは、リトリバ内のウィンドウ4508をはぎ取る(tear off)ことができる。これを行うためには、ユーザーはLH3の割当を受け、CSLにスキーマの巡航を要求しなければならない。

本発明には、知識ベースのクライアント手段と知識ベースのサーバ手段が含まれる場合がある。知識ベース・サーバ手段は、オブジェクト指向のロック・マネ

ジャ手段によって構成されていることが望ましい。知識ベース・サーバ手段には、ダイナミック・クラス・マネジャ手段、コネクション・マネジャ手段、問い合わせマネジャ手段、ハンドル・マネジャ手段、ユニット・マネジャ手段、データベース・マネジャ手段およびファイル・マネジャ手段が含まれていることが望ましい。

図258は、知識ベース・サーバ4132のための情報処理・通信環境を提供するコンピュータ・ハードウェア構成4109の主要構成部分を示したものである。この構成は、メインメモリ6101からプログラム命令を呼び出し、実行する演算論値ユニット6100を含む、中央処理装置、すなわちCPU5109から構成されている。プログラムは、ディスクドライブ4110に記憶されており、プログラムへのアクセスはディスクコントローラ6106によって行われる。知識ベース・ファイルもディスクドライブ4110に記憶されており、これへのアクセスはメインメモリ6101の仮想メモリアドレスを通じて行われる。この仮想メモリアドレスを通じて、要求された場合にディスク・ファイル6108の連続データのページ6111がメインメモリ6101にコピーされる。本発明の好適実施例では、この知識ベース管理システムに仮想メモリ6112が使用されている。知識ベース・サーバ4102は、ネットワーク・コントローラ6102によってアクセスが制御されるローカル・エリア・ネットワーク4100か、シリアル・インターフェース・コントローラ6103を通じてアクセスが制御されるワイド・エリア・ネットワーク6104を通じて、クライアントAP1413とインタ

ラクトする。I/Oバス6105は、CPU6109と周辺データ・ストレージ、インターフェースおよび通信構成部品との間のデータ転送を仲介する。

図259は、リトリーバ4130、スキーマ・エディタ4144、レガシー4133のグラフィカル・ユーザー・インターフェース構成部品およびAPI4143のための情報処理・通信環境を提供するコンピュータ・ハードウェア構成4112の主要構成部分を示したものである。この構成は、メインメモリ2601からプログラム命令を呼び出し、実行する演算論値ユニット6100を含む、中央処理装置、すなわちCPU6109から構成されている。プログラムは、単数または複数のディスクドライブ6110に記憶されており、プログラムへのアクセスはディスクコントローラ6106によって行われる。ユーザーは、CRTディスプレイ4113上に表示されるグラフィカル・ユーザ

ー・インターフェースとキーボード4115およびマウスまたは同様のグラフィカル・ポインター4114によって、システムとインタラクトする。API4143は、ネットワーク・コントローラ6102によってアクセスが制御されるローカル・エリア・ネットワーク4100か、シリアル・インターフェース・コントローラ6103を通じてアクセスが制御されるワイド・エリア・ネットワーク6104を通じて、知識ベース・サーバ4132とコミュニケーションする。I/Oバス6105は、CPU6109と周辺データ・ストレージ、インターフェースおよび通信構成部品との間のデータ転送を仲介する。

本発明は、図204で示したような知識ベース・クライアントと知識ベース・サーバで構成されるクライアント/サーバ構成でより有利に利用することができる。しかし、本発明は、必ずしもクライアント/サーバ構成には限定されない。本発明は、分散型データベース・システムでも利用することができる。

C. オブジェクト指向データベースの構造

図267は、パーツを編集するさいに取る手順を示すフローチャートを示したものである。たとえば、図266を参照すると、パーツを編集するアクセス権を持つユーザーは、エディット・ボタン7180を作動させて、図292に示すようにパーツ・ウィンドウ5019をオープンすることができる。図367に示すように、最初のステップ5012は、ユーザーがパーツ・エディタ・ウィンドウ5019からアトリビュー

トおよび編集するパーツを選択することにかかわるものである。アトリビュート5101に関して新たな値あるいは改訂値5061を入力すると、システムはステップ5013において、パラメータの入力を受け入れる。このアトリビュートが列挙されたアトリビュート5101である場合には、図293に示すように、ユーザーが選択できるプルダウン・リスト5062が表示される。図267のステップ5014においては、もっと多くのパーツを編集するかどうかの決定を行う。それ以上編集するパーツが存在しなければ、フローはステップ5017に進む。システムは、パーツ・ディスプレイ5020とパーツ・エディタ・ウィンドウ5019を、エディットされた値5061によって更新する。次にシステムはステップ5018に進み、コントロールをユーザーに返す。

ステップ5014において、まだ編集するパーツが残っている場合には、フローはステップ5015に進み、システムは次の選択されたパーツを取り上げる。ステップ5016において、システムは、次の選択されたパーツのパラメータをユーザー入力値5061にセットする。コントロールは次に、ステップ5014にループバックする。

図294は、パーツを削除する手順を示すものである。ステップ5021において、削除するパーツをパーツ・エディタ・ウィンドウ5019から選択する。次に、パーツ削除コマンドボタン5026をクリックする。ステップ5022においては、まだ削除するパーツが残っているかどうかの判定を行う。答がイエスの場合には、フローはステップ5023に進み、ここでシステムは次の選択パーツを取り上げ、照会結果と知識ベースからこのパーツを削除する。次にフローは、ステップ5022にループバックする。削除するパーツがなくなると、システムはステップ5024に進み、更新された照会結果をエディタ・ウィンドウ5019に再表示する。次にフローはステップ5025に進み、コントロールをユーザーに返す。

図295は、パーツを移動させる手順のフローチャートを描いたものである。この手順は、ステップ5102に示すように、パーツ・エディタ・ウィンドウ5019から移動させるパーツを選択することによって開始することができる。あるいはこの手順は、ステップ5103において、パーツ・エディタ・ウィンドウ5019上でクラスの階層をナビゲートして移動先クラスを選択することにより、開始することがで

きる。ユーザーは、たとえば図284に図解するように、パーツ移動コマンド・ボタン5027を作用させることができる。

図295を参照すると、手順はステップ5104に進み、まだ移動させるパーツが残っているかどうかの判定を行う。これ以上移動させるパーツが存在しなければ、フローはステップ5042に移り、システムは、更新された照会結果をエディタ・ウィンドウ5019に再表示する。次にフローはステップ5043に進み、コントロールをユーザーに返す。

図295のステップ5104に戻って、まだ移動させるパーツが存在するという判定を行った場合には、フローはステップ5105に進み、システムは次の選択パーツを取り上げる。ステップ5106においては、ユーザーが無条件の移動を要求している

かどうかの判定を行う。答がイエスの場合には、フローはステップ5040にジャンプする。システムは次に、パーツのクラスを、ユーザーが選択した移動先のクラスに設定する。何らかのパラメータあるいは失われたアトリビュートは、未定義に設定される。フローはステップ5041に進み、システムは移動させたパーツを照会結果から削除する。フローは、ステップ5042に進み、ここでシステムは、照会結果をエディタ・ウィンドウ5019に再表示する。

ステップ5106において、無条件の移動を要求しなかった場合には、フローはステップ5107に進み、そこで何らかのパーツのパラメータのアトリビュートが移動先クラスから失われているかどうかについての判定を行う。答がノーであれば、フローはステップ5040に進み、上記の手順を継続する。

ステップ5107において、移動先クラスから失われたパーツ・パラメータのアトリビュートが存在するという判定が行われた場合には、フローはステップ5108に移る。システムは、移動によって削除されるパラメータのリストを取って、それをディスプレイ4116上に表示することによりユーザーに提示する。次にフローは、ステップ5109に進む。そこでユーザーが、パラメータが削除されるとの警告を無視するか、あるいはパーツを無条件に移動させるように要求すると、フローはステップ5040に移り、上記の手順を継続する。パラメータ削除の警告を無視しなかった場合、あるいはパーツを無条件に移動させるよう要求しなかった場合には

、フローはステップ5104にループバックする。

パーツを編集する過程は、パーツ・エディタ・ウィンドウ5019の説明（図284参照）との関連において理解を深めることができる。クラス7174およびサブクラス7196、7197、7198、7199を選択することによってパーツを指定し、アトリビュート検索基準7177を入力し、表示順序4194を設定した後、ユーザーはエディット・コマンド・ボタン7180を選択して、パーツを編集することができるようになる。このコマンド7180を選択すると、パーツ・エディタ・ウィンドウ5019が表示される。パーツ・エディタ・ウィンドウ5019の上部エリア5102には、クラス・ツリー5044が表示され、編集しているパーツのナビゲーション・パスとクラス定義を強調表示して示す。パーツ・エディタ・ウィンドウ5019の下部エリア5103は、編集するパーツ5020が表示される。パーツは、スプレッドシート・アプリケーション

ンに使用される表と同様な表の表5020に表示される。パーツ・アトリビュート5049、5100、5101、等々とアトリビュート値5105、5106、5107、等々は、パーツ仕様ウィンドウ7170においてユーザーがあらかじめ設定しておいた表示順序で、左から右に表示される。数値を使用するためには、入力ボックス5063をクリックする。新しい数値をキャンセルするためには、キャンセル・ボックス5064をクリックする。

パーツ・エディタ・ウィンドウ5019の上部エリア5102には、クラス定義5044が表示される。このエリアは、編集するために選択したパーツのナビゲーション・パスとクラス定義を示すクラス・ツリーを構成する。ウィンドウ5019には、水平分割バー1047があり、ウィンドウを2つの区分5102と5103に分割している。この分割バー5047は、上下に動かして、区分5102を大きくして見たり、他方の区分5103を大きくして見たりすることができる。パーツ・エディタ・ウィンドウ5019は、編集エリア5046として参照できるエリアを含んでいる。アトリビュート値5101を選択した後は、テキスト・ボックスあるいはリスト・ボックス5104がこのエディット・エリア5046に表示され、変更が加えられるようになる（図292参照）。各パーツは、表5020中の列5048に表示され、表5020中の列5048の各列には番号がつけられる。この列番号は、ユーザーが情報を必要とするパーツ、あるいは移動

させたり削除したいパーツを選択するために使用することができる。アトリビュート5049、5100、5101、等々はカラム見出しであり、アトリビュート値は列である。

ユーザーが新しいパーツを知識ベースに入れることを決定した後は、ユーザーはそのパーツを完全に指定しなければならない。好適実施例においては、完全なパーツ仕様は、リーフ・クラス7201までのクラスを選択して、必要なすべてのアトリビュート7203の数値を入力することによって定義される。好適実施例においては、リーフ・クラス7201を選択せず、あるいは必要なアトリビュート7203を入力した場合には、パーツを追加することはできない。パーツを作るときに望ましい手順は、可能なかぎり完全なパーツ仕様を指定するためにできるだけ多くのアトリビュート値7203を入力することである。

パーツが追加できるようになるまでに、いくつかのアトリビュートが必要であ

る。パーツ作成コマンド7181を選択する前に、要求されるアトリビュートそれぞれについてアトリビュート値を入力しなければならない。また、保護されているアトリビュートに関しては、アトリビュート値を入力することはできない。保護されているアトリビュートには、アトリビュートのアイコンのすぐ左に、保護アイコン7191がつけられている。リーフ・クラス7201を選択して、要求されたすべてのアトリビュートを入力すると、パーツ作成コマンド・ボタン7181を選択できるようになる。パーツ作成コマンド7181を選択すると、そのパーツはユーザーの知識ベースに追加され、見つけられたパーツ7172が更新されて4001のパーツ・カウントが表示される。

知識ベース・クライアント4131は、C++ライブラリのセットであり、API 4143を通じてクライアント・アプリケーション4131、4133、4144に知識ベース・サービスを提供する。このサービスは、ローカルなものとしてもでき、あるいは知識ベース・サーバ4132へのリモート・プロシージャ・コールとすることもできる。ウィンドウズで使用するアプリケーションの場合は、この知識ベース・クライアントは、1つまたは2つ以上のウィンドウズ・ダイナミック・リンク・ライブラリ（DLL）によって構成され、これらのDLLはWinSockのDLLを使用して

、知識ベース・サーバ4132とレジストリ・サーバ4141へのネットワークアクセスを提供する。

知識ベース・サーバ4132は、UNIX サーバ・プロセスで、知識ベース4110のアクセス、検索、更新を管理する。1つまたは2つ以上の知識ベース4110および4110を管理する。

ダイナミック・クラス・マネージャ4134は、知識ベース・サーバ4132のソフトウェア・サブシステムで、スキーマとデータを管理する。ダイナミック・クラス・マネージャ4134は、動的に修正することのできるクラス、アトリビュート、単位、インスタンスの知識を記憶する能力を持っている。ダイナミック・クラス・マネージャ4134は、C++のライブラリとクラスによって構成され、「リーガサイジング」(legacizing)のオペレーションと、クラス、アトリビュート、インスタンス、パラメータ、単位ファミリー、単位、メタアトリビュートのランタイムにおけるアクセス、作成、削除、修正のオペレーションを行う。

ダイナミック・クラス・マネージャ4134の機能には、ユーザー・プログラムにより、API 4143が提供する一連の関数を通じてアクセスする。

ダイナミック・クラス・マネージャ4134の知識ベース（以下単に「知識ベース」と呼ぶこともある）は、クラス、アトリビュート、単位、パラメータ値つきインスタンス、これらのオブジェクトの関係の集合である。ダイナミック・クラス・マネージャ4134においては、クラスは別々の種類のオブジェクトを定義する。各クラスはすでに、アトリビュートを定義済みである。アトリビュートにはいくつかの種類があり、オブジェクトの特性を定義するために役立つものである。クラスは、別のクラスから生成させることができる。その場合には、そのクラスはもとのクラスからのアトリビュートを引き継ぐ。知識ベースは、クラスのインスタンスを格納している。インスタンスによって定義されるアトリビュート値は、パラメータである。

クラス、インスタンス、アトリビュート、パラメータの概念を説明するもうひとつの方法は、犬を例に取ることである。「犬」という語は、クラスの類似語である。犬とは、一連の特性あるいはアトリビュートを持つ同様なもののグループ

である。犬のアトリビュートは、色、品種、名前などのことである。クラスとアトリビュートは、いかなる特定の犬も表現しないが、犬を表現するための便宜を与える。犬のインスタンスには、アトリビュートに値を与えるパラメータがつけられる。たとえば、色がベージュで品種がゴールデンリトリバーで名前がサンダーという犬である。

クラスは、関係を持たせることができる。クラス「犬」は、もっと大きいクラス「哺乳類」の部分である。クラス「哺乳類」は、「犬」よりも具体性が小さい。クラス「哺乳類」は、オブジェクト「犬」についての情報をクラス「犬」ほど詳しく伝えないが、「哺乳類」についてのあらゆることは「犬」にも当てはまる。「犬」は、明らかに「哺乳類」のサブセットであり、この関係はサブクラスである。「犬」は「哺乳類」のサブクラスである。サブクラス「犬」はさらに、大きな「犬」、小さな「犬」、等々のサブクラスに小分類される。サブクラスという概念は、この2つのクラスの親子関係を含むものである。「哺乳類」は親であり、「犬」はサブクラスである。「『犬』は『哺乳類』から派生する」という語

法も、この関係を説明するために使われる。

サブクラス「犬」は、親のクラス「哺乳類」のアトリビュートを引き継ぐ。アトリビュート「色」は、「哺乳類」のクラスのアトリビュートでもありうる。すべての「哺乳類」には色があるからである。「犬」のクラスは、アトリビュート「色」を親から引き継いでいるのである。

ルートクラスは特別である。このクラスには親がない。ルートクラスは、すべてのクラスが派生しはじめるルート（根元の）クラスである。ここに提示する図解においては、クラスの階層を説明するためにグラフが描かれており、その図の最上部にルート・クラスが置かれている。各サブクラスは、ルート・クラスからずっと広がっていくパスに分岐し、そのためにこのグラフは、逆さの樹のように見える。クラスのグループ全体は1本の樹であり、親のないこの特別クラスは、最上部にあっても根（ルート）である。

ダイナミック・クラス・マネージャ4134がサポートして使用できるアトリビュートの型の1つは、数値型である。数値型のアトリビュートは、現実の世界の

計測可能な数量を表現するために使用される。このような測定値は、数値だけで構成されるものではない。これらの測定値には、いくつかの関連したユニット（単位）も持っている。ダイナミック・クラス・マネージャ4134は、ユニット・マネージャ4138と共同で、数値型のアトリビュートで利用できる各種のユニットについての情報を維持する。ダイナミッククラスマネージャ4134（ユニット・マネージャ4138を使用する）は、ユニット間の換算が必要な場合にはその変換を行う。システムが理解するユニットは、各ユニット・ファミリーに分類される。これらのユニット・ファミリーとユニット・ファミリーが定義するユニットは、ランタイム中に変更することができる。ダイナミック・クラス・マネージャ4134はまた、ダイナミックユニット・マネージャ4138を構成する。

「スキーマ」という語は、クラス、アトリビュート、ユニット、ユニット・ファミリーのレイアウトを指す。インスタンスのない知識ベースは、スキーマである。このことは、ダイナミック・クラス・マネージャ4134が管理する各種のオブジェクトについての後述の詳細な説明との関連で、理解を深めることとなる。クラスは、本発明のスキーマにおいて、最も基本的なオブジェクトである。クラ

スとは、関連オブジェクトの集合である。この例においては、1つのクラスは8個または9個の構成要素を取りうる。クラスは、スキーマのオブジェクトである。上記に説明したように、スキーマは、クラス、アトリビュート、ユニット、ユニット・ファミリーとその関係の集合である。あらゆるクラスは、ルート・クラス7173を除いて、1つだけの派生した親を持っている。ルート・クラス7173は、親を持たないただ1つのクラスである。ルート・クラス7173は、決して削除することができないという点で、もう1つの特別な特性を持っている。親から派生するクラスの帰結は、そのクラスが親の属性をすべて持っているということの意味する。これらの属性は、アトリビュートと呼ばれる。アトリビュートは、親のクラスから引き継がれたものである。

クラスは、ゼロあるいは1つ以上のサブクラスを持つことができる。クラスは、そのサブクラスそれぞれの親である。サブクラスとは、親を持つクラスであり、したがってルートクラス7173は、サブクラスではない。親のクラスのサブクラ

スには、ある種の定義された順序がある。この順序は、一定している。つまり、ダイナミック・クラス・マネージャ4134は、知識ベースを終了して再起動したときにもこの順序を維持する。

クラスは、そのサブクラス、サブクラスのサブクラス、等々のすべてによって構成される一連の子孫を持っている。サブクラスがゼロであるかあるいは子孫のセットが空のクラスは、リーフ・クラス7201と呼ばれる。サブツリーとは、クラスとその子孫すべてによって構成されるセットである。サブツリーは、そのクラスに根付いていると呼ばれる。サブクラスはまた、1セットの祖先を持っており、このセットは、ルート・クラス7173を含む親、親の親、等々によって構成されるセットである。同じ親を持つクラスは、ときには兄弟と呼ばれることがある。

サブクラスを親までたどることは、ツリーを登ると呼ばれることがある。親からサブクラスの1つに移動することは、ツリーを降りると呼ばれることがある。したがって、スキーマのルート・クラス7173は、いちばん上、ツリーのてっぺんにあり、いちばん下、ツリーのルート・オブジェクトは、代表的にはリーフ・クラス7201である。

クラスには名前があり、この名前は、クラス、サブクラス、またはリーフ・ク

ラスを識別するテキストで、ASCII文字列である。本発明は、クラスの参照のためにクラス・ハンドルを使用する。クラス・ハンドルの詳細については、ハンドル・マネージャ7137との関連において後述する。図264に示す例においては、3つのサブクラスがある。

図285は、クラス4800の内部オブジェクトの表現を示すものである。このスキーマにおいては、クラスには親のハンドル4801がつけられている。各クラス・オブジェクト4800は、親がないルート・クラス7173を除いて、その親クラスのハンドルを示す情報が記憶されている。このクラスのこの場所には、ヌルが記憶されている。ハンドルは、オブジェクトの参照(reference)である。親のハンドル情報4801は、ハンドル・マネージャ7137が、そのクラスの親クラスである記憶されたクラス・オブジェクト4800を識別するために使用する。

クラス・オブジェクト4800は、サブクラス・リスト4802を格納している。サブ

クラス・リスト4802は、ハンドルの配列であり、これは、ハンドル・マネージャ7137がそのクラスのサブクラスであるクラス・オブジェクト4800を識別するために使用することができる。本発明においてもたらされる内部表現においては、リストは無限に、動的に成長する。利用できる記憶空間は固定されていない。

このことは、データベースの構造に柔軟性とパワーを与える。クラス・オブジェクト4800は、実質的に性能を劣化させることなく、大きなデータベースにきわめて多数のサブクラスを記憶できるからである。

クラス・オブジェクト4800は、アトリビュート・リスト4803を格納している。ハンドル・マネージャ7137は、アトリビュート・リスト4110に記憶された情報を使用して、クラス・オブジェクト4800が保有するアトリビュートを識別することができる。

クラス・オブジェクト4800はまた、ローカル・インスタンス・リスト4804を格納している。これはハンドル・リストである。図285に示すフィールド4805は、クラス名すなわちクラスを識別するテキストの記憶位置を指すポインタである。

フィールド4806は、クラス4800のハンドルを記憶するために使用される。フィールド4807は、クラス・コードすなわちそのクラスが一次クラスであるか二次クラスであるか集合であるかの表示(indication)を記憶する。

クラス・オブジェクト4800はまた、サブツリー・インスタンス・カウンタ4808を格納する。サブツリー・インスタンス・カウンタは、クラス4800のすべての子孫に存在する項目あるいはインスタンスの総数、すなわちすべてがクラス4800のサブクラスであり、すべてがクラス4800のサブクラスのサブクラスであり、等々のクラス4800のインスタンスの総数である。したがって、サブクラスを選択し、オープンして知識ベースのツリー構造をナビゲートすると、ツリーのどの位置においても、現行のクラスのサブツリー・カウンタ4808を検索してパーツ数が直ちに知らされ、その情報はリトリバ4130に渡される。サブツリー・インスタンス・カウンタ4808は、知識ベースが修正されるときにつねに更新され、ユーザーがデータベースのツリー構造をナビゲートするさいに、必ずしもパーツ発見数7172のリアルタイム計算を行う必要がない。

また図285を参照すると、クラスオブジェクト4800はまた都合よくメタパラメータ・リスト4809を格納している。このメタパラメータ・リスト4809は、文字列であり、たとえばクラス4800が表示するパーツの種類のグラフィック・ディスプレイを格納するファイルの名前、データのリーガサイズに使用するシソーラス情報、あるいはその他のリーガサイズ情報などの関連情報を格納する文字列のポインタとして使用することができる。

図286は、総称リスト4810の例を示すものである。クラス・マネージャ4134は、変動量のデータがオブジェクトと関連づけられる場合にはつねに、ハンドルのリスト、浮動小数点値のリスト、文字列を特徴づけるポインタのリスト、等々を使用する。リストの例は、項目4802、4803、4804、4809となる。リスト4810は、単純な整数のリストを示す。

リスト・オブジェクト4810は、リスト・データ4811の先頭4815を指すポインタ4812を含んでいる。リスト・オブジェクト4810はまた、現在リスト・データ4811に割り当てられたサイズを示すフィールド4813を含んでいる。リスト・オブジェクト4810はまた、現在使用しているリスト・データ4811の量を示す情報を格納するフィールド4814を含んでいる。

リスト・データ4811は、実際の数値のリストを格納している。この例におけるリストの最初の項目4815は、数値「4005」を含んでいる。同様に、この例のリス

トの項目4816、4817、4819、4820、4821は、追加の数値を格納している。この例におけるリスト項目4822、4823、4824、4825、4826は、現在使用されておらず、ゼロに設定されている。この図示例においては、現在割り当てられているリストのサイズは、12である。このリストの4814で使用されている量は7であり、これはリストの最初の7項目が有効であることを意味する。

図287は、アトリビュート・データ4827のデータ構造を図解したものである。アトリビュート・オブジェクト4827は、図示した実施例において、少なくとも6つのフィールドを含んでいる。最初のフィールド4828は、そのアトリビュートの名前であるASCII文字列を構成する外部名を指すポインタを含んでいる。アトリビュート・オブジェクト4827はまた、このアトリビュート・オブジェクト4827の

ハンドルを格納するフィールド4829を含んでいる。アトリビュート・オブジェクト4827はまた、このアトリビュート4827を定義するクラスのハンドルを格納するフィールド4830を含んでいる。第4のフィールド4831は、このアトリビュートがそのクラスを定義するために必要とされるアトリビュートかどうかについてのブール標識である。第5のフィールド4832は、このアトリビュートが保護されているかどうかを示すブールフィールドを含んでいる。このことは、保護アイコン7191によって示される。図287において示されるアトリビュート・オブジェクト4827のデータ構造においては、この情報はフィールド4832に記憶される。アトリビュート・オブジェクト4827はまた、メタパラメータ・リストであるフィールド4833を含んでいる。

列挙されるアトリビュートには、集合的にアトリビュート・データ4834として表示されるフィールド4828～4833と、列挙子(enumerator)ハンドルのリストであるフィールド4835がある。

ブール・アトリビュートの場合には、フィールド4828～4833だけが使用される。これらのアトリビュートも同じく、図287においては集合的に示されている。

数値アトリビュートは、アトリビュート・データ4834として集合的に示されるフィールド4828～4833と、この数値アトリビュートのユニット・ファミリーのハンドルを格納するフィールド4838を含んでいる。

文字列アトリビュートの場合、ならびに文字列配列アトリビュートの場合には

、フィールド4828～4833を構成するアトリビュート・データ4834だけが含まれる。

これらのデータ構造をダイナミック・クラス・マネージャ4134が使用する1つの例は、クラスと関連するクローズ・フォルダ・アイコンをクリックしてクラスを選択する手順である。クラスがオープンされると、ダイナミック・クラス・マネージャ4134は、クラス・オブジェクト4800をチェックし、アトリビュート・リスト4803を検索する。アトリビュート・リストに記憶されているハンドルは、ハンドル・マネージャ7137に渡される。ハンドル・マネージャ7137は、そのクラスの各アトリビュートの仮想メモリ・アドレスを返す。ダイナミック・クラス・マネージャ41

34は、次に、アトリビュート・オブジェクト4827の外部名を指すポインタ4828を使用して、そのアトリビュートの外部名の文字列テキストを検索する。このASCII テキスト情報は、次に、API 4143に渡され、そこからこの情報は最終的に、リトリバ4130に送られて、ディスプレイ4116上に表示される。

図288は、列挙子オブジェクト4841のデータ構造を示すものである。列挙子4841は、3つのフィールドを構成することができる。最初のフィールド4842は、列挙子オブジェクト4841の外部名を指すポインタを含んでいる。第2のフィールド4843は、列挙子オブジェクト4841のハンドルを含んでいる。第3のフィールド4844は、メタパラメータ・リストを格納することができる。ハンドルは、他のオブジェクトから列挙子オブジェクトへのリンクを形成するために使用される。この構造の長所は、オブジェクトの外部名を変更することが望ましい場合に容易に知識ベースを修正することができる能力である。このような変更は、外部名を表示するために使用されるASCII文字列に関して1回だけ行う必要がある。他のすべてのオブジェクトは、ダイナミック・クラス・マネージャ4134に実際の外部名を与えるためにハンドル・マネージャ7137を使用することのできるハンドルを格納しているだけである。

図289は、インスタンス4871と関連パラメータ4872のデータ構造を示すものである。インスタンス・オブジェクト4971は、4つのフィールド4873なども4876を含むことができる。最初のフィールド4873は、このインスタンスの所有クラスのハンドルである。第2のフィールド4874は、その所有クラスのインスタンス・リス

ト4804の中にこのインスタンスのハンドルの元の位置を与えることができる。第3のフィールド4875は、パラメータのリストであり、これは4877に格納される数値を指示するものである。第4のフィールド4876は、インスタンス・オブジェクト4871のハンドルである。パラメータ・リスト4877は、このインスタンス・オブジェクト4871と関連する各種のアトリビュートのパラメータに対する複数のポインタを含んでいる。図289に図解した例においては、リスト4877には3つのエントリー4878、4879、4880が含まれている。リスト4877の追加要素は、明確化のために省略してある。リスト4877中のポインタ4878は、関連パラメータに関する情報

を指示する。パラメータ4872のデータ構造は、図290において詳細に図解する。

図290は、5つの異なる型、すなわち列挙、ブール、数値、文字列、文字列配列のそれぞれの型のパラメータのデータ構造を示すものである。このパラメータオブジェクト4872のそれぞれは、アトリビュート・ハンドル4881を持っている。列挙されたオブジェクト4888は、アトリビュート・ハンドル4881と列挙子ハンドル4882を持っている。ブール・オブジェクト4889は、アトリビュートハンドル4881とブール値4883を持っている。数値パラメータ・オブジェクト4890は、アトリビュート・ハンドル4881、ユニット・ハンドル4884、数値4885を持っている。たとえば、数値パラメータが4010オームであり、ユニット・ハンドル4884がオームのユニットのハンドルであるとするれば、数値4885は4010となる。文字列パラメータ4891は、アトリビュート・ハンドルのフィールド4881と、ASCII文字列に対するポインタ4886を含んでいる。文字列配列パラメータ4892は、アトリビュート・ハンドル4881と、文字列配列に対するポインタのリストを指示するフィールド4887を含んでいる。

図291は、インスタンスを持つスキーマの例である。この例には、「エレクトロニクス」と名づけられたクラスがあり、このクラスには「コンデンサ」と名づけられたサブクラス4800'がある。コンデンサのサブクラス4800'は、「ケース型」と呼ばれるアトリビュート4827を持っている。この例におけるケースには、2つの可能な型があり、「ケースA」とおよび「ケースB」と呼ばれている。サブクラスコンデンサ4800'は、「電解」と名づけられたサブクラス4800'がある。電解サブクラス4800'は、「電圧定格」と呼ばれるアトリビュート4827'を持っ

ており、それぞれ5ボルトとA型Bケースのパラメータ4890と4888を持つ1つのインスタンス4871'が与えられている。ほとんどのオブジェクトとリストは図解を単純化するために不完全なものとして表現されているが、同様の参照数値は、図273～278との関連で記述されるものと同じオブジェクトを指している。

図291において、クラスオブジェクト4800には、名前4806があり、この名前はこの例においては「エレクトロニクス」である。このクラスオブジェクト4800にはフィールド4802があり、このフィールドはサブクラスのリスト4893を指示する

。リスト4893には最初のエントリ4894があり、このエントリはサブクラス4800'のハンドルである。この例においては、サブクラス4800'の名前4806'はコンデンサである。もちろん、スキーマオブジェクトに対するすべての参照は、実際にはハンドル（図291には示していない）を使用し、実際にはハンドルマネージャ7137を経由してハンドル・テーブルを使用する。このことは、ダイアグラムを単純化するために、図291には示していない。

サブクラス4800'のコンデンサには、フィールド4802があり、このフィールドは、サブクラス4893'のリストを指示する。リスト4893'にはエントリ4894'があり、このエントリはサブクラス4800'のハンドルである。サブクラス4800'の名前4806'は電解である。サブクラス4800"には、フィールド4802'にヌルのエントリがあり、ここには通常、サブクラスのリストがあればそのリストのポイントが置かれている。この例においては、サブクラス4800"は、サブクラスを持っていない。

コンデンサ・サブクラス4800'に戻って、フィールド4803には、アトリビュート4897のリストのポイントがある。リスト4897には、「ケース型」と呼ばれる列挙されたアトリビュート4827がある。列挙されたアトリビュート・オブジェクト4827のフィールド4830には、コンデンサと呼ばれる定義サブクラス4800'のハンドルがある。列挙されたアトリビュートオブジェクトには、列挙子のハンドルのリスト4839を指示するポイント4835がある。この例においては、リスト4839には、列挙子4841のハンドル4839が含まれている。列挙子4841は、この列挙子の外部名を指示するポイント4842を持っており、このポイントは、「ケースA」のASCII文字列とすることができる。同様に、リスト4839における項目4899は、ケー

スBと関連する列挙子4841'を指示する。

ここで、電解と名づけられたサブクラス4800'に戻ると、ポイント4803'はアトリビュートのリスト4897'を指示し、リスト4897'の中の1つのフィールドには「電圧定格」である数値アトリビュート4827'のハンドルが含まれている。数値アトリビュート4827'にはフィールド4830'があり、このフィールドにはクラスを定義するハンドルが含まれている。このハンドルはこの例においては電解と

名づけられたクラス4800” 数値アトリビュートオブジェクト4827’にはまた、フィールド4838’があり、このフィールドには電圧ユニット・ファミリーのハンドルが含まれている（図示せず）。

電解クラス4800’に戻ると、フィールド4804”には、インスタンスのハンドルのリスト4895のポインタがある。リスト4895中の項目4897には、インスタンス4871と関連するハンドルがある。インスタンス4871にはフィールド4873があり、このフィールドは、所有クラスのハンドルがある。このハンドルは、この例においては電解クラス4800”である。インスタンス・データ・オブジェクト4871にはまた、フィールド4875があり、このフィールドはパラメータ4877のリストを指示する。リスト4877にはポインタ4878があり、このポインタは数値パラメータ4890を指示する。数値パラメータ4890にはフィールド4881があり、このフィールドはアトリビュート4827のハンドル（電圧定格）がある。数値パラメータ4890にはまた、フィールド4884があり、このフィールドにはユニットのハンドルが含まれている。このハンドルは、この例においては「ボルト」である。数値パラメータオブジェクト4890には、フィールド4885があり、このフィールドには数値5.0が置かれている。この例においては、電解コンデンサは、5.0ボルトの定格である。

パラメータ・リスト4877は、ポインタ4879があり、このポインタは列挙されたパラメータ4888を指示する。数値パラメータ・オブジェクト4888にはフィールド4881’があり、このフィールドにはアトリビュートのハンドルが置かれている。このアトリビュートはこの例においてはケース型である。数値パラメータ・オブジェクト4888にはまた、フィールド4882があり、このフィールドには列挙子4841’のハンドルがある。この例においては、定格5.0ボルトの電解コンデンサは、ケースB型である。

ここで説明するデータ構造には、大きな長所がある。図291を参照すれば明らかのように、このデータ構造においては、名前や摘要を変更することが容易である。データベースが、Bケース型の1,000件のコンデンサのインスタンスを格納している例を考えてみよう。Bケース型を中断するか、あるいは「強化」(re-enforced)に変更する場合に、行う必要のある唯一の変更は、ケース型の名前を示

すASCII文字列を1つだけ置き換えることである。データベースの1,000件のインスタンスはすべて、ハンドルマネージャがこのASCII文字列と関連づけるハンドルを格納しているだけである。その他のいかなる変更も、このデータベースにおいては行う必要がない。

本発明によるデータ構造のもう一つの長所は、一次的な値が未定義の場合には何も記憶されないということである。したがって、むだに使用される空間は皆無である。

このデータ構造のもう一つの長所は、アルゴリズムをツリー構造の位置に応じて変更する必要がないということである。すべてのアルゴリズムは、ツリー構造中の位置にかわりなく、同じように機能する。唯一の特殊な場合は、ルート・クラスである。たとえば、1つのインスタンスをデータベースに追加するアルゴリズムは、位置しているツリー構造のどこにおいても同じである。このことは、スキーマの動的変更をきわめて簡単なものにする。ツリー構造中のあるクラスまたはブランチ全体が、単にハンドルのリストを変更するだけで、1つの位置から別の位置に移動させることができる。変換プログラムを起動する必要はない。あらゆるものが内蔵されている。クラス・オブジェクト4800は、その親のハンドルを含んでおり、したがって、その親が何であるかがわかる。クラス・オブジェクト4800はまた、そのサブクラスのリストのポインタ4802を持っており、したがってその子が何であるかがわかる。

このデータベース構造においては、インスタンスをすばやく削除することができ。インスタンスは、インスタンス4804のリスト中の最後の項目を取り出して、それを削除するインスタンスの位置に移動させることによって削除することができる。言い換えれば、最後のインスタンスのハンドルが、削除されるインスタンスのハンドルの上に書き込まれて、このリスト中の項目の数が1つ減少することになる。

このデータベース構造においては、インスタンスをすばやく削除することができ。インスタンスは、インスタンス4804のリスト中の最後の項目を取り出して、それを削除するインスタンスの位置に移動させることによって削除することができる。言い換えれば、最後のインスタンスのハンドルが、削除されるインスタンスのハンドルの上に書き込まれて、このリスト中の項目の数が1つ減少することになる。

とになる。インスタンス・オブジェクト4874のインスタンス・インデックス・フイルド4874は、高速の削除を容易にするために使用することができる。

好適実施例においては、パラメータの値はつねにベースのユニットに記憶される。記述したフィールド中のオブジェクトは、必ずしも1ワードのメモリを占有

する必要はない。好適実施例においては、特定の型のすべてのパラメータが連続的に記憶される。このことは、検索の速度を向上させる。たとえば、図291への参照によって記述したケース型4841'は、他のすべてのケース型のパラメータと一緒に連続的に記憶させることができる。5.0ボルトという数値パラメータは、他の数値型のボルトパラメータとともに、連続したメモリの異なった物理位置に記憶される。

上述のとおり、クラスオブジェクト構造4800にそのクラスのサブツリー・インスタンス・カウントを行うフィールド4808を与えることは、システムが仮想的にパーツ・カウント7172を表示して、ユーザー検索のツリー遷行ステップ中にユーザーが瞬間的なフィードバックを行うことを可能にする。パーツを見つけるプロセスは、基本的に、希望したアトリビュートを持っていない数千個のパーツを捨てて、検索を少数にしばることとなる。

このことは、分類階層のルートから正しいクラスまでをナビゲートすることによって達成される。この段階において、パーツ発見指示7172は、サブツリー・インスタンス・カウントを示すデータ構造フィールド808を使用して更新される。このことは、各ステップごとに使用できるインスタンスを実際に計数することに比べて、著しい応答時間の短縮となる。ユーザーは、選択したツリー中で使用できるパーツの数を示す瞬間的なフィードバックを行えるのである。オブジェクト指向階層ツリーの組み合わせは、何らかの希望するアトリビュートの組み合わせに基づく検索基準の組み合わせとあいまって、現行の検索基準およびクラスに該当するインスタンスの数に関するフィードバックを可能にする一方、従来試みられてきたデータベース管理スキーマに対する大きな長所を実現している。

ダイナミック・クラス・マネージャ4134の1つの重要な機能は、オペレーション中にデータベース構造を変更できる能力である。このデータベース構造は、スキーマと呼ばれる。オブジェクト指向データベースのスキーマは、クラスを使用し

て構造化される。このクラスは、アトリビュートを含んでいる。このアトリビュートは、列挙子とユニット・ファミリーを含むことができる。これらの項目を追加し、移動させ、削除する能力は、データベースの動的なオペレーションにとつ

て重要なものである。

クラスをスキーマに追加するためには、3つの項目が既知でなければならない。すなわち、クラス名、新しいクラスの親、サブクラスのリスト中の新しいクラスを挿入する位置、である。図292は、このオペレーションを図解するものである。最初のステップ5840は、親クラスのハンドルを実際のクラス・ポインタに変換する。この親のポインタは、使用する前に直ちにステップ5841においてテストしなければならない。このポインタが無効であると判明した場合には、このオペレーションはステップ5842で終了する。ポインタが有効であれば、ステップ4843において挿入インデックスをテストする。このインデックスが無効であると判明した場合には、オペレーションはステップ5844で終了する。最後に、ステップ5845においてクラス名のテストを行って、このクラス名が有効なクラス名のガイドンスに適合しているかどうかを判定しなければならない。このクラス名が失敗した場合には、オペレーションはステップ5846で終了する。ステップ5845がこのクラス名を受け入れた場合には、新しいクラスを生成することができる。新しいハンドルがまずステップ5847で生成され、次に新しいクラスがステップ5848において内部メモリ中に生成される。この新しいハンドルは、図293のステップ5849においてクラス・ハンドルの表に挿入され、次にステップ5850において、このハンドルはサブクラス・ハンドルの親のリストに追加される。この最後のオペレーションは、ファイル・マネージャ4140に、この新しいクラスを二次記憶装置4110上の指定した親に追加させることとなる。

アトリビュートをクラスに追加するためには、3つの項目が既知でなければならない。すなわち、所有クラスのクラス・ハンドル、新しいアトリビュートを挿入する位置、アトリビュートの名前、である。図294は、このアトリビュートの追加を図解するものである。最初のステップ5930は、クラス・ハンドルをクラス・ポインタに変換することであり、次にステップ5931においてこのクラス・ポインタが有効なクラス・ポインタであるかどうかをテストする。このクラスポイン

タが有効でなければ、この手順はステップ5932で終了する。クラス・ポインタが有効であると判定された場合には、その挿入インデックスはステップ4933におい

で有効性が確認される。このインデックスが有効性テストに失敗した場合には、この手順はステップ5934で終了する。このインデックスが有効性テストに合格した場合には、オペレーションはステップ5935に進み、ここでアトリビュートの名前がテストされる。アトリビュート名がテストに失敗した場合には、オペレーションはステップ5936で終了する。アトリビュート名がステップ5935で受け入れられた場合には、そのアトリビュートを生成することができる。ステップ5937では、このアトリビュートの新しいハンドルが生成される。次に新しいアトリビュートが、5939において、所有クラスに対してローカルなアトリビュートのリストに挿入される。最後のステップである図295のステップ5940においては、ファイルマネージャ4140に、この新しいアトリビュートを二次記憶装置4110上の指示された親に追加させることとなる。このオペレーションは、ステップ5941で完了する。

インスタンスの追加は、図284に示すとおりである。インスタンスの追加には、クラス・ハンドルが必要である。このクラス・ハンドルは、ステップ5918において、クラスポインタに変換しなければならない。このクラス・ポインタは、ステップ5931において、有効なクラス・ポインタであるかどうかをテストする。このクラス・ポインタが有効でなければ、この手順は5920で終了する。クラス・ポインタが有効であると判定された場合には、手順はステップ5921に進み、そこで新しいインスタンス・ハンドルと新しいインスタンス・オブジェクトが生成される。新しいインスタンスのハンドルは、5922において、ハンドル・テーブルに挿入される。このインスタンスは、5923において、インスタンスの親のリストに挿入される。5924において、そのサブツリー・インスタンス・カウントは、新しいインスタンスの存在を反映して増加する。このインスタンスはここですでにメモリ中に生成されており、二次記憶装置4110に追加する必要がある。この追加は、図285のステップ5925において行われる。この手順は、ステップ5926で完了する。

。 クラスの削除は、図286に示すとおりである。クラスをデータ構造から削除するためには、現行のクラス・ハンドルを識別しなければならない。このクラス

・ハンドルは、まずステップ6600において、クラス・ポインタにデコードしなけ

ればならない。このクラス・ポインタは次に、ステップ6601において、有効なクラス・ポインタであるかどうかをテストする。このクラス・ポインタが無効である場合には、この手順は6602で終了する。クラス・ポインタが有効であれば、6603において、このクラス・ポインタがチェックされて、このクラスがルート・クラスであるかどうか判定される。このクラス・ポインタがルート・クラスを表示するものである場合には、ルート・クラスは削除できないため、手順は6604で終了する。クラス・ポインタがルート・クラスを表示していなければ、6605において、このクラス・ポインタがチェックされて、このクラスがリーフ・クラスであるかどうか判定される。クラス・ポインタがリーフ・クラスを表示していない場合には、手順は6604で終了する。クラス・ポインタがリーフ・クラスを指示していることが判明した場合には、オペレーションはステップ6906に進み、ここでこのクラスのインスタンスのすべてが削除される。インスタンスを削除する過程は、図290を参照しながら後述する。ステップ6607においては、削除されるクラスに対してローカルであるアトリビュートのすべてが削除される。図287において、このクラスは次にステップ6608において、その親クラスとのリンクが切り離される。システムは、6609において、このリンク切り離しが成功したかどうか、そのクラス・リストを格納するデータ構造がかん全体残されているかどうかを判定するチェックを行う。リンクの切り離しが失敗した場合には、オペレーションはステップ6610で終了する。リンクの切り離しが成功した場合には、オペレーションはステップ6611に進み、そこでクラスオブジェクトが実際に削除される。ステップ6612において、ファイルマネージャ4140は、このクラスオブジェクトを二次記憶装置4110から削除するよう命令され、オペレーションはステップ6613で完了する。

アトリビュートの削除は、図288に示すとおりである。アトリビュートを削除するためには、ステップ5860において、アトリビュートのハンドルをアトリビュート・ポインタにデコードしなければならない。ステップ5861は、ステップ5860によって得られたアトリビュート・ポインタが有効であるかどうかをチェックする。このアトリビュート・ポインタが無効である場合には、この手順は5862で終

了する。アトリビュート・ポインタが有効であれば、手順は5863に進み、このアトリビュートから派生するサブツリーのすべてのインスタンスのすべてのパラメータ全体を検索する。この検索後ステップ5864において、システムは、どれだけ多くのパラメータがこのアトリビュートから派生したものであるかを判定する。このアトリビュートから派生したパラメータが存在した場合には、オペレーションは5865に進むが、ここでパラメータは未定義とされる。このアトリビュートから派生したパラメータが存在しなかった場合には、手順はステップ5866に進む。同様に、1865においてパラメータが未定義とされた後、オペレーションは5866に進む。ステップ5866において、アトリビュートは定義クラスからリンクを切り離される。5867において、手順は、このリンク切り離しのオペレーションが成功したかどうかを判定するチェックを行う。リンクの切り離しが失敗した場合には、オペレーションはステップ5868で終了する。リンクの切り離しが成功した場合には、図289の5869において、アトリビュート・オブジェクトが削除される。次にステップ5870において、ファイルマネージャ4140は、このアトリビュート・オブジェクトを二次記憶装置4110から削除するよう命令される。オペレーションはステップ5871で完了する。

インスタンスの削除は、図290に示すとおりである。インスタンスは、まずステップ6000において、インスタンスのハンドルをインスタンス・ポインタに変換してデータベースから削除する。このインスタンス・ポインタは、ステップ6001において、このポインタが実際に有効なインスタンス・ポインタであるかどうかを判定するチェックを行う。このインスタンス・ポインタが無効である場合には、このオペレーションは6002で終了する。インスタンス・ポインタが有効であれば、このインスタンスは6003において、所有クラスからリンクを切り離される。このインスタンス・オブジェクトは、6004において、自己を削除する。次に6005において、サブツリー・インスタンス・カウントは、1つのインスタンスがそのサブツリーから削除されたことを示して減少する。次にステップ6006において、ファイル・マネージャ4140は、このインスタンスの削除を反映してこのインスタンス・オブジェクトを二次記憶装置4110から削除するよう命令される。オペレーションはステップ6007で完了する。

図291においては、クラス階層におけるサブツリーの新しい位置への移動を説明する。ステップ5800において、サブツリー移動手順は、移動させるクラス、移動先親クラス、指定移動先におけるサブリング・クラス間の位置によって呼ばれる。ステップ5801においては、移動させるクラスのクラス、ポイントと、移動先の親クラスが得られる。ステップ5802においてすべての有効なポイントについてのテストが失敗した場合には、ステップ5804はエラーを返し、そうでなければ、そのクラスが自明のこととして親クラスに移動することを防止するためにテスト5805を行う。ステップ5806は、移動先親クラスのサブクラス間の位置が有効範囲内にあり、エラーが生じた場合にはステップ5804がエラーを返すことを保障する。ステップ5807においては、移動させるクラスと移動先のクラスの双方の上のクラス階層が分析され、最も近い共通の祖先のクラスが識別される。

図292のステップ5808においては、この共通の祖先が、移動させるクラスと同一であるかどうかをテストする。同一であれば、そのクラスが親に移動させようとしているのではないことを保障するためのテストがすでに行われているのであるから、次にこの移動は、クラスを自分のサブクラスに移動させる試みであると判定されて、エラーが返される。その他のすべての移動は適法のものであり、したがってそのクラスはステップ5809において、親クラスからのフックを外されて、ステップ5810において、移動先クラスのサブクラスのリストに追加される。ステップ5811において、移動先のクラスのサブツリー・カウントは移動させるクラスのインスタンスの数だけが増加し、ステップ5812において、元の親のサブツリー・カウントは移動させるクラスのインスタンスの数だけが減少する。ステップ5813において、知識ベースのパラメータ画像は、ファイル・マネージャ4140によって更新されて、ステップ5814でコーラーへの復帰を成功させる。

図293は、移動させるクラスのその元の親クラスからのフックを外す過程を説明するものである。ステップ5816においては親のクラス・ポイントが得られ、このポイントはステップ5817において親クラスのサブクラスのリストを得るために使用される。ステップ5817において、その結果移動させるクラスのハンドルがテストされたとおり得られない場合には、その知識ベースは内部的に一貫性がなく、エラーがコーラーに返される。そうでない場合には、ステップ5818においてそ

のクラスは親クラスのサブクラス・リストから削除され、その後ステップ5819において復帰を成功させる。

図294は、移動させるクラスと移動先のクラスとの最も近い共通の祖先を発見する過程を示すものである。ステップ5820において、一時的なクラスハンドルが移動させるクラスのハンドルに設定される。ステップ5821では、その一時的なクラスの親を見つけて、移動させるクラスからルートクラスへの移動の順序でクラスのリストを生成するループを開始する。ここで遭遇する各クラスは、ステップ5222においてリストに追加され、ステップ5223においてルートが見つめられると反復過程は終了する。ステップ5823におけるテストが失敗した場合には、ステップ5824において一時的なクラス・ハンドルが親クラスのハンドルに設定され、反復過程は継続される。

同様なリストが、ステップ5828から5831までにおいて移動先クラスに関して生成され、図295に進む。ステップ5831においては、一時的なクラスハンドルが移動先クラスのハンドルに設定される。ステップ5832では、その一時的なクラスの親を見つけて、移動させるクラスからルート・クラスへの移動の順序でクラスのリストを生成するループを開始する。ここで遭遇する各クラスは、ステップ5826においてリストに追加され、ステップ5227においてルートが見つめられると反復過程は終了する。ステップ5827におけるテストが失敗した場合には、ステップ5828において一時的なクラス・ハンドルが親クラスのハンドルに設定され、反復過程は継続される。

最後のステップ5829は、この結果得られた2つのリストを、適合するクラス・ハンドルが見つかるまで反復する。これは、最も近い共通の祖先であり、ステップ5830において返される。

D. アトリビュート値によるインスタンスの比較

検索あるいは照会を行うための望ましい方法および装置については、1994年10月10日出願の出願番号No. 08/339, 481に詳細に記述してある。検索の結果が得られると、そのインスタンス（図解例においては、インスタンスはパーツである）は、図262Aに示すように表示することができる。これらのパーツは次に、それぞ

れのアトリビュート値によって比較される。ユーザーが比較しようと望むパーツは、その上をクリックして選択する。選択が行われると、選択されたパーツのディスプレイは陰をつけて表示されるか、あるいは強調表示され、図262Aに示す例においては参照数値4653～4662によって指示する陰をつけたパーツのディスプレイが表示されている。パーツ4663、4664、その後のパーツは、図解例において選択されていないため、強調表示されない。

検索仕様（図262A参照）に適合するパーツのリストを表示させた後、ユーザーは多くの場合、これらのパーツをその共有するアトリビュート値によって比較する。このことは、アクション・メニュー4651からパーツ比較オプション4652を使用して行うことができる。このコマンド4652は、図262Bおよび図263に示すパーツ・アトリビュート比較ダイアログボックスにアクセスするコマンドであり、ここでユーザーは、すべての選択したパーツ46334634、4635、4636の間でアトリビュート値を比較することができる。好適実施例においては、ユーザーは、パーツ比較コマンド4652を呼び出す前に、検索結果ウィンドウ4650において少なくとも2つ以上のパーツを選択していなければならない。選択したパーツのアトリビュートを他のすべての値と比較する例を、図262Bに示すパーツ・アトリビュート比較ダイアログ・ボックスにおいて掲げる。

好適実施例においては、図262Bに示すようにパーツ・アトリビュート比較ダイアログ・ボックス4630がディスプレイに最初に表示される前に、選択したパーツのすべてのアトリビュート値が、同じ値を持つか持たないかに関して評価される。パーツ・アトリビュート比較ダイアログ・ボックス4630がディスプレイに画面に表示されるさいに、この比較の結果は、ダイアログ・ボックス4630の最初のカラム4637に示される。図解した例においては、等号演算子（＝）632が最初のカラム4637に表示され、ここで、列4643のすべてのアトリビュートが、パーツ比較オペレーションのために選択したパーツ4633、4634、4635、4636のすべてに関して等しくなる。不等号（<>）演算子4631は、最初のカラム4637に表示され、ここで、列4642のすべてのアトリビュートが、パーツ比較オペレーションのために選択したパーツすべてに関して等しくないものとなる。ダイアログの第2カラム4638は、アトリビュートのタイトルをリスト表示するものであり、残りのカラム

4633、4634、4635、4636は、それぞれ1つずつのパーツ、すなわち、ユーザーがあらかじめ検索結果ウィンドウから選択したパーツの1つずつに割り当てられる。各パーツ・カラム4633、4634、4635、4636は、他のカラムと同じ順序で、それぞれのアトリビュート値をリスト表示する。

パーツ・アトリビュート比較ダイアログ・ボックス4630の例を、図262Bに掲げる。表13は、パーツ・アトリビュート比較ダイアログ・ボックス4630の領域を示すものである。

表 1 3

領域	摘 要
初期評価 4637	パーツアトリビュート比較ダイアログボックス4630がディスプレイに最初に表示されるときに、等号(=)演算子あるいは不等号(<>)演算子を表示する。等号演算子4632は、その特定のアトリビュートのすべての値が選択したパーツすべてに関して等しいことを示す。不等号演算子4631は、同一のアトリビュートすべてに関して少なくとも1つの値が選択したパーツすべてに関して等しくないことを示す。
アトリビュートタイトル 4638	各アトリビュートの名前を別々の列に表示する。
パーツ4633、 4634 4635、 4636	特定のパーツに関する各アトリビュートの値を表示する。各アトリビュート値は該パーツに対応するカラムに関する別々の列の1つの要素である。パーツ(カラム)番号は、各パーツに対応するカラムの最上部にある。

図262Bおよび図263を参照すると、特定のコマンドボタン4639、4640、4641が

、図示した実施例に設定されている。「選択したパーツの比較」のコマンドボタ

ン4639は、システムに、ダイアログボックス4630に表示される他のパーツ4633、4634、4636のすべてのアトリビュート値を、ユーザーが選択した1つだけのパーツに属するアトリビュート値と比較させる（図263参照）。ユーザーは、パーツ4635を、そのカラム番号4635（図4060において「パーツ4003」とラベル表示されている）をクリックして選択しなければならない。「比較クリア」のコマンド・ボタン4640は、システムに、「選択したパーツの比較」のコマンドボタン4639を使用して比較が行われた後比較結果をクリアさせる（この時点でディスプレイは、図262Bに示したと同様なディスプレイに復帰する）。「クローズ」コマンド・ボタン4641は、システムが、パーツ・アトリビュート比較ダイアログ・ボックスあるいはウィンドウ4630をクローズして、パーツ比較ダイアログ・ボックス4630がオープンされる前にアクティブだったディスプレイ・ウィンドウに復帰させる。表14は、パーツ・アトリビュート比較ダイアログ・ボックス4630のコマンド・ボタン4639、4640、4641を示すものである。

表 1 4

コマンド	摘 要
選択 パーツ 4639 と比較せよ	ダイアログ・ボックスに表示されるすべてのアトリビュート値をユーザーが選択した1つだけのパーツに属するアトリビュート値と比較する。ユーザーは、このコマンドを選択する前にパーツ（すなわちカラム番号を選択しなければならない）。
比較 4639 を クリアせよ	選択 パーツ 比較 コマンドをあらかじめ比較を行った後に、比較の結果をクリアする。
ク ロ ー ズ 4641	ダイアログ・ボックスをクローズする。

図263を参照すると、選択したパーツを比較するコマンドを出すと、アトリビュートディスプレイが変化して、比較の結果を、等しいか等しくないかの比較を

直ちに見られるようにするというきわめて便利な方法で示す。選択しなかったパーツ4633、4634、4636のすべてのアトリビュート値が基準パーツ4635のアトリビュート値と比較されるさいに、合致するアトリビュート値のセル4644と4645は、陰をつけられずに表示される。たとえば、選択された、あるいは基準のパーツ4635は、アトリビュート「主要素材」4648の値を持ち、そのパーツが「鋼」4646製であることを示す。選択されたパーツのアトリビュート値「鋼」4646は、他のパーツの値とアトリビュート「主要素材」に関して比較される。最初のパーツ4633は、このアトリビュート4648に関して「鋼」4644の値を持っている。これは、選択されたパーツ4635のアトリビュート値「鋼」4646と同じ値4644であるため、図263に示すように陰をつけずに表示される。同様に、第2のパーツ4634も、このアトリビュート4648に関して「鋼」4645の値を持っている。これは、選択されたパーツ4635のアトリビュート値「鋼」4646と同じ値4645であるため、これも図263に示すように陰をつけずに表示される。第4のパーツ636は、このアトリビュート4648に関して「ナイロン」4647の値を持っている。これは、選択されたパーツ4635のアトリビュート値4646とは同じ値でもひとしくもないため、図263に示すように陰をつけたセル4647として表示される。

部品アトリビュートを比較する手順は、次のステップを含むものである。

1. 検索結果ウィンドウから、比較したいと望む2つ以上のパーツを選択する。

2. アクションメニューから、比較パーツを選択する。パーツ・アトリビュート比較ダイアログ・ボックス4630が表示され、1つのアトリビュートに関するアトリビュート値が等しい（＝）4632か、あるいは1つのアトリビュートに関するアトリビュート値が異なっている（＜＞）4631かを、最初のコラム4637に表示する。

3. 図263を参照して、ダイアログ・ボックス4630に表示されるすべてのパーツ4633、4

634、4636のアトリビュート値を、基準パーツ4635のアトリビュート値と比較するために、基準パーツ・コラム番号4635をクリックして、選択パーツ比較コマン

ド・ボタン4639を選択する。選択されなかったパーツ4633、4634、4636のすべてのアトリビュート値が、基準パーツ4635のアトリビュート値と比較される。一致したアトリビュート値のセル4644と4645は、陰がつけられない。一致しないアトリビュート値のセル4647は、陰をつけて表示される。

4. 比較の結果をクリアするために、比較クリアのコマンド・ボタン4640を選択する。

5. ダイアログに表示されるすべてのパーツのアトリビュート値を異なった部品のアトリビュート値と比較するために、ステップ4003を繰り返す。図260と図261は、パーツのアトリビュートを比較する過程のフローチャートを描いたものである。ステップ4625において、比較のために複数のパーツを選択する。もちろん、1つのパーツしか選択しなかった場合には基準パーツと比較すべきものが何もないことになるため、複数のパーツを選択しなければならないのである。ステップ4626においては、アクションメニュー4651から、パーツ比較コマンド4652を選択する。

ステップ4627において、ウィンドウあるいはダイアログ・ボックスがオープンされ、比較のために選択されたパーツが表示される。図262Aにおいて、パーツ・アトリビュートは、列で表示されることが望まれている。好適実施例においては、図262Bに示すように、パーツ・アトリビュートは、行で表示されることが望まれている。次に、ステップ4628において、比較するパーツ4635を選択する。参照数4628で識別された点Aは、図260および図261のフローチャートにおける共通の点である。

ステップ4630は、外部プログラム・ループへのエントリ点であり、ステップ4631は、内部プログラム・ループへのエントリ点である。ステップ4632において、システムは現行のインスタンスが選択された基準インスタンス4635であるかどうかを判定するチェックを行う。これがそのとおりであれば、このメソッドはステップ4635にジャンプし、次のインスタンスあるいはカラムに進む。そうでない場合には、メソッドはステップ4663に進み、ここでメソッドは、対応するアトリビ

ュート値が現行のインスタンスと選択されたインスタンス4635に関して同じであ

る（あるいは一致している）かどうかを判定する。このアトリビュート値が等しい場合には、アトリビュート列4648のセル4644のディスプレイは変わらず、フローはステップ4635に進んで、そこで次のインスタンスに進む。アトリビュート値が等しくない場合には、メソッドはステップ4634に進み、アトリビュート列4648のセル4647のディスプレイは変更されて、たとえば強調表示されたり、背景色が変更されたり、陰がつけられたりする。フローは次にステップ4635に進んで、次のインスタンスあるいはカラムに進む。ステップ4636においては、これがこのアトリビュートの最後のインスタンスであるかどうか、すなわちこれが最終カラムであるかどうかを判定するチェックが行われる。これがこのアトリビュートの最後のインスタンス、すなわち最終カラムである場合には、手順は続行して次のアトリビュートの比較を行う。すなわち、次の列に進む。ステップ4637においては、これが最終列であるかどうかを判定するチェックを行う。そうでない場合には、プロセスはステップ4630にループバックする。これが最終列であれば、比較はすべての列およびカラムに関して完了している。すなわち、各アトリビュートは、あらゆるインスタンスに関して比較されている。次にシステムは、ステップ4638において終了する。

E. 要約

本発明を、パーツ情報を管理するオブジェクト指向データベース管理システムの使用にかかわる望ましいアプリケーションにおいて使用することにより、多数のユーザーが同じ知識ベースに同時にアクセスして、パーツを探し、パーツを編集し、スキーマを編集することができる。このオブジェクト指向データベース管理システムは、「ロック」を使用することによって同時性を管理する。

複数のスキーマ・エディタあるいはデベロッパ4144が、同じ知識ベースにおいて同時にアクティブとなることができる。ユーザーが編集したいと望むクラスを選択すると、スキーマ・エディタ4144は、そのクラスにロックを設定する。そのスキーマ・エディタ4144がそのクラスにロックを設定しているかぎり、そのクラスとすべてのサブクラスは、他のいかなるスキーマ・エディタ144での編集のた

めにアクセスすることはできず、リトリバ4310によって見るように利用するこ

ともできない。しかし、別のスキーマ・エディタ4144および/またはリトリーバ4130は、この知識ベース4123のロックを設定されていない他のいかなるセクションにおいても同時に作業を行うことができる。

スキーマ・デベロッパ/リトリーバの同時性によって、ユーザーは自分のスキーマ4123を、このオブジェクト指向データベース管理システムを使用してパーツ情報を検索している会社の他のユーザーと同時に編集することができる。ロックされている領域においてパーツを検索あるいは編集しようとする者は誰でも、そのクラスがロックされていることを示す情報を受け取れることが望ましい。このメッセージが表示されると、最初のユーザーは、知識ベース4123の他の領域に行くか、あるいは第2のユーザーのスキーマ・エディタ4144がそのロックを解除するまで待つことができる。

編集機能はすべて、ロックホルダーとなる請求を必要とし、次にロックの書きを要求して、その後はじめて編集を成功させることができる。

F. ソフトウェアの関数

列挙関数pmx_lockTypeは、知識ベースにおいて要求され、解除することのできるロックタイプを指定するために使用される。

```
typedef enum {  
    PMX_ERROR_LOCKTYPE      =0,  
    PMX_NO_LOCK              =1,  
    PMX_CLASS_S_LOCK         =2,  
    PMX_TREE_S_LOCK          =3,  
    PMX_TREE_U_LOCK          =4,  
    PMX_TREE_X_LOCK          =5  
} pmx_lockType;
```

列挙関数pmx_lockModelは、知識ベースにおけるクラスのロック状態を記述するためにために使用される。与えられたいかなるクラスも、そのクラスに存在するロックによって、そのクラスに関して明示的あるいはそのクラスがロック

されるサブツリーの中にあることによって定義される何らかのロック状態にある

。

```
typedef enum {
    PMX__LOCKMODE__ERROR    =0,
    PMX__LOCKMODE__NONE     =1,
    PMX__LOCKMODE__SHARE    =2,
    PMX__LOCKMODE__UPDATE   =3,
    PMX__LOCKMODE__EXCLUSIVE =4,
} pmx__lockMode;          =5
```

ロックについての情報を返すAPI関数`pmx__getLockDescriptor()`によって返される

`pmx__lockDescriptor`の構造は、指定したクラスの指定したロック・ホルダによって保持される。指定したクラスとロック・ホルダは、各タイプのロックが獲得した回数とともに返される。

```
typedef struct {
    pmx__classHandle      classHandle;
    pmx__lockHolderHandle lockHolderHandle;
    long                  classShareLockCount;
    long                  treeShareLockCount;
    long                  treeUpdateLockCount;
    long                  treeExclusiveLockCount;
} pmx__lockDescriptor;
```

以下のAPI関数は、コンカレンシー・コントロール同時制御のために提供されることが望ましい。

```
pmx__startLockHolder
pmx__endLockHolder
pmx__requestLock

pmx__releaseLock
pmx__releaseAllLocks
```

```

pmx__releaseAllLocksOfType
pmx__freeLockDescriptor
pmx__getLockDescriptor
pmx__getLockMode
pmx__equalLockHolderHandles
pmx__isNullLockHolderHandle
pmx__getNullLockHolderHandle

```

これらの関数は、ロック・ホルダとなることを開始、終了し、ロックを要求、解除し、クラスのロック状態についての情報を検索するために使用される。

ロック・ホルダは、ロック・ホルダ・ハンドルによって識別され、`pmx__startLockHolder()`によって開始され、`pmx__endLockHolder()`によって終了する。

ロックを要求するためには、`pmx__requestLock`を使用する。ロックあるいはロックのグループを解除するためには、`pmx__releaseLock()`、`pmx__releaseAllLocks()`、または

`pmx__releaseAllLocksOfType()`を使用する。

クラスに関して取得しているロックについての情報を検索するためには、`pmx__getLockMode()`または`pmx__getLockDescriptor()`を使用する。

以下に、各関数についての説明を掲げる。

<code>whichDB</code>	オープンされた知識ベースのハンドル。
<code>lockHolder</code>	起動されているロック・ホルダのハンドル。
<code>thisClass</code>	ロック情報が要求されているクラスのクラス。

説明

この関数は、所定のロック・ホルダおよびクラスについて取得した各タイプの計数値を返す。レポートされるのは、所定のクラスについて要求されているロックだけである。

クラスは、祖先に対するツリーのロックによって影響を受けることがあるが、この条件はレポートされない。請求は、終了したときにその記述子を解除しなければならない。請求はまた、`pmx__freeLockDescriptor()`関数がデータは破壊さ

れないものと予定しているため、記述子のいかなるデータも変更あるいは破壊しないよう注意しなければならない。

リターン値

1. pmx__endLockHolder

目的

起動しているロック・ホルダを終了する。

構文

```
cd__boolean  
pmx__endLockHolder(  
    pmx__dbHandle          whichDB,  
    pmx__lockHolderHandle  lockHolder);
```

パラメータ

whichDB オープンされている知識ベースのハンドル。

lockHolder 起動しているロック・ホルダのハンドル。

説明

ロック・ホルダが終了する。このロック・ホルダ・ハンドルによって要求されたいかなるロックも、自動的に解除される。この関数は、ロック・ホルダ・ハンドルが無効な場合（たとえば、ロック・ホルダが起動していなかった、など）には失敗する。

リターン値

成功したときには、CD_TRUEを返す。

失敗したときには、CD_FALSEを返す。

エラー

PMX__ERRORBADDBHANDLE

この知識ベースのハンドルは無効である。

PMX__ERRORBADLOCKHOLDERHANDLE

このロック・ホルダのハンドルは無効である。

2. pmx__getLockDestructor

目的

所定のクラスに維持されているロックの記述を得る。

構文

```
pmx__lockDescriptor CD_FAR*  
pmx__getLockDescriptor(  
    pmx__dbHandle      whichDB,  
    pmx__lockHolderHandle lockHolder,  
    pmx__classHandle    thisClass);
```

パラメータ

[リターン値]

成功したときには、記述子を指すポインタを返す。

失敗したときには、NULLのポインタを返す。

エラー

PMX__ERRORBADCLASSHANDLE

このクラスハンドルは無効である。

PMX__ERRORBADDEHANDLE

この知識ベースのハンドルは無効である。

PMX__ERRORBADLOCKHOLDERHANDLE

このロック・ホルダのハンドルは無効である。

3. pmx__getLockMode

目的

所定のクラスのロック・モードを返す。

構文

```
pmx__lockMode  
  
pmx__getLockMode(  
    pmx__dbHandle      whichDB,  
    pmx__lockHolderHandle lockHolder,  
    pmx__classHandle    thisClass,
```

```

        cd_boolean                self);
パラメータ
    whichDB                      オープンされている知識ベースのハンドル。
    lockHolder                   起動しているロック・ホルダのハンドル。
    thisClass                    ロック・モードを要求するクラスのハンドル。
    self                         現行の請求 (self) あるいはその他のすべての請求に
                                関して、どのロックモードを要求するかを指定する。

```

説明

この関数は、所定のクラスのロック・モードを返す。このロックモードは、そのクラスとそのクラスの祖先におけるロックによって、そのクラスの効果的なロックとなる。請求は、取得しているロックに基づくモードを要求するか、あるいは他の請求によって保持されているロックに基づくモードを要求するかが選択できる。selfの引数がCD_TRUEである場合には、そのロック・モードの結果は、現行の請求とロック・ホルダーが取得しているロックに基づくものとなる。そうではなく、selfがCD_FALSEである場合には、そのロック・モードの結果は、他のすべての請求とロック・ホルダーに基づくものとなる。

リターン値

成功したときには、そのロック・モードを返す。

失敗したときには、PMX_LOCKMODE_ERRORを返す。

エラー

PMX_ERRORBADBOOLEANVALUE

ブール値がCD_TRUEでもCD_FALSEでもない。

PMX_ERRORBADCLASSHANDLE

クラス・ハンドルが無効である。

PMX_ERRORBADBHANDLE

知識ベースのハンドルが無効である。

PMX_ERRORBADLOCKHOLDERHANDLE

ロック・ホルダのハンドルが無効である。

4. pmx_releaseAllLocks

目的

1つのクラスとその子孫すべてに関して取得したロックすべてを解除する。

構文

```
cd_boolean  
pmx_releaseAllLocks(  
    pmx_dbHandle          whichDB,  
    pmx_lockHolderHandle  lockHolder,  
    pmx_classHandle       thisClass):
```

パラメータ

whichDB	オープンされている知識ベースのハンドル。
lockHolder	起動しているロック・ホルダのハンドル。
thisClass	ロックを解除しようとするサブツリーのルートにあるクラスのハンドル。

説明

この関数は、所定のクラスをルートとするサブツリーにおけるすべてのクラスに関して保持しているすべてのロックを解除する。解除されるのは、与えられたロック・ホルダが解除されたロックだけである。ロックがまったく取得されていない場合には、エラーは生じない。

リターン値

成功したときには、CD_TRUEを返す。

失敗したときには、CD_FALSEを返す。

エラー

PMX__ERRORBADCLASSHANDLE

クラス・ハンドルが無効である。

PMX__ERRORBADDBHANDLE

知識ベースのハンドルが無効である。

PMX__ERRORBADLOCKHOLDERHANDLE

ロック・ホルダのハンドルが無効である。

5. pmx__releaseAllLockOfType

目的

所定のクラスをルートとするサブツリーにおけるすべてのクラスに関して取得した所定のタイプのロックすべてを解除する。

構文

```
cd__boolean  
pmx__releaseAllLocksOfType(  
    pmx__dbHandle          whichDB,  
    pmx__lockHolderHandle  lockHolder,  
    pmx__classHandle       thisClass,  
    pmx__lockType          lockType);
```

パラメータ

whichDB	オープンされている知識ベースのハンドル。
lockHolder	起動しているロック・ホルダのハンドル。
thisClass	ロックを解除しようとするサブツリーのルートにあるクラスのハンドル。
lockType	解除しようとするロックのハンドル。

説明

この関数は、所定のクラスをルートとするサブツリーにおけるすべてのクラスに関して保持しているすべてのロックを解除する。解除されるのは、与えられたロック・ホルダが解除されたロックだけである。ロックがまったく取得されていない場合には、エラーは生じない。

リターン値

成功したときには、CD_TRUEを返す。

失敗したときには、CD_FALSEを返す。

エラー

PMX__ERRORBADCLASSHANDLE

クラスハンドルが無効である。

PMX__ERRORBADDBHANDLE

知識ベースのハンドルが無効である。

PMX__ERRORBADLOCKHOLDERHANDLE

ロック・ホルダのハンドルが無効である。

PMX__ERRORBADLOCKTYPE

ロックタイプが無効である。

6. pmx__releaseLock

目的

所定のクラスに関して取得したロックすべてを解除する。

構文

```
cd_boolean  
pmx__releaseLock(  
    pmx__dbHandle      whichDB,  
    pmx__lockHolderHandle lockHolder,  
    pmx__classHandle    thisClass,  
    pmx__lockType       lockType);
```

パラメータ

whichDB	オープンされている知識ベースのハンドル。
lockHolder	起動しているロック・ホルダのハンドル。
thisClass	ロックを解除しようとするクラスのハンドル。
lockType	解除しようとするロックのハンドル。

説明

この関数は、所定のクラスとロック・ホルダの所定のタイプの1つのロックを解除する。請求は、1つのクラスの同一のタイプに関して複数のロックを取得できるため、ロックの解除はそのつど請求して何度も行わなければならない。ロックは、
pmx__releaseAllLocksによって一括して解除することができる。

この関数は、ロック・ホルダ、クラス・ハンドル、ロック・タイプによって記述されたロックが取得されていない場合には、エラーを生じない。

リターン値

成功したときには、CD_TRUEを返す。

失敗したときには、CD_FALSEを返す。

エラー

PMX__ERRORBADCLASSHANDLE

クラス・ハンドルが無効である。

PMX__ERRORBADDBHANDLE

知識ベースのハンドルが無効である。

PMX__ERRORBADLOCKHOLDERHANDLE

ロック・ホルダーのハンドルが無効である。

PMX__ERRORBADLOCKTYPE

ロック・タイプが無効である。

PMX__ERRORNOSUCHLOCK

解除しようとしたロックが存在しない。

7. pmx__requestLock

目的

所定のクラスに関して取得したロックすべてを解除する。

構文

```
cd__boolean  
pmx__requestLock(  
    pmx__dbHandle          whichDB,  
  
    pmx__lockHolderHandle  lockHolder,  
    pmx__classHandle       thisClass,  
    pmx__lockType          lockType);
```

パラメータ

whichDB オープンされている知識ベースのハンドル。

lockHolder	起動しているロック・ホルダのハンドル。
thisClass	ロックを要求するクラスのハンドル。
lockType	要求するロックのハンドル。

説明

この関数は、所定のクラスとロック・ホルダの所定のタイプの1つのロックを要求する。このロックは、その要求が他の請求およびロック・ホルダが保持しているロックと衝突しなければ取得することができる。

リターン値

成功したときには、CD_TRUEを返す。

失敗したときには、CD_FALSEを返す。

エラー

PMX__ERRORBADCLASSHANDLE

クラス・ハンドルが無効である。

PMX__ERRORBADDBHANDLE

知識ベースのハンドルが無効である。

PMX__ERRORBADLOCKHOLDERHANDLE

ロック・ホルダのハンドルが無効である。

PMX__ERRORBADLOCKTYPE

ロック・タイプが無効である。

PMX__ERRORCANNOTGRANTLOCK

要求したロックが許可されない。

8. pmx__startLockHolder

目的

新しいロック・ホルダとなる。

構文

```
pmx__lockHolderHandle
pmx__startLockHolder(
    pmx__dbHandle          whichDB);
```

パラメータ

whichDB オープンされている知識ベースのハンドル。

説明

この関数は、ロック・ホルダ・ハンドルによって識別される新しいロック・ホルダを生成する。この新しいロック・ホルダは、ロックを要求するために使用することができる。

同一の請求においても、あるロック・ホルダからのロックが他のロックと衝突することがある。

リターン値

成功したときには、新しいロック・ホルダのハンドルを返す。失敗したときには、NULLのロック・ホルダ・ハンドルであるpmx_NullLockHolderを返す。

エラー

PMX__ERRORBADCLASSHANDLE
クラス・ハンドルが無効である。

9. pmx__freeLockDescriptor

目的

pmx__freeLockDescriptorを解放する。

構文

```
cd__boolean  
pmx__freeLockDescriptor  
pmx__LockDescriptor(  
    pmx__LockDescriptor*thisDescriptor);
```

パラメータ

thisDescriptor オープンされている知識ベースのハンドル。

説明

この関数は、ロック記述子と関連するメモリを解放する。

この関数を呼び出した後は、もうこの記述子を参照することはできない。

リターン値

成功したときには、CD_TRUEを返す。
失敗したときには、CD_FALSEを返す。

エラー

PMX__ERRORNULLPOINTER

要求した入力引数の代わりにNULLのポインタが渡される。

10. pmx__equalLockHolderHandles

目的

2つのロック・ホルダを、相等性に関して比較する。

構文

```
cd_boolean  
pmx__equalLockHolderHandles(  
    pmx__LockHolderHandles(  
        pmx__LockHolderHandle    handle1  
        pmx__LockHolderHandle    handle2);
```

リターン値

2つのハンドルが等しい場合には、CD_TRUEを返す。
等しくない場合には、CD_FALSEを返す。

11. pxm__NullLockHolderHandle

目的

NULLのロック・ホルダ・ハンドルを得る。

構文

```
pxm__NullLockHolderHandle  
pxm__getNullLockHolderHandle();
```

リターン値

NULLのロック・ホルダ・ハンドルを返す。

12. pxm__isNullLockHolderHandle

目的

ロック・ホルダ・ハンドルがNULLのハンドルであるかどうかをチェック

する。

構文

```
cd_ boolean  
pdx__isNullLockHolderHandle(  
    pmx__lockHolderHandle);
```

リターン値

指定したハンドルがNULLのロック・ホルダ・ハンドルである場合には、CD_ TRUEを返す。

失敗した場合には、CD_ FALSEを返す。

上記の説明は、本発明の1例を挙げることをのみを目的として、現在の好適実施例を説明するものにすぎない。改良および代替的な実施例は、本開示の利点を理解することができれば、当業者にとっては明白なものであろう。本発明の範囲は、本特許明細書に記述した特定の例にかぎられるべきものではなく、本発明の範囲は、特許請求範囲によって定義されるものである。

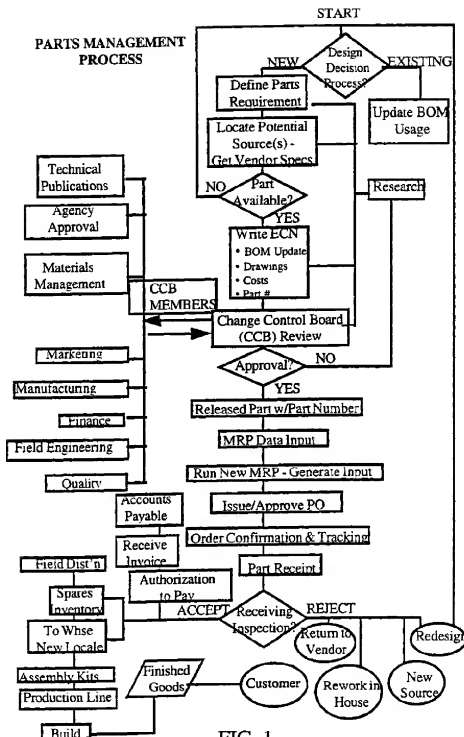


FIG. 1

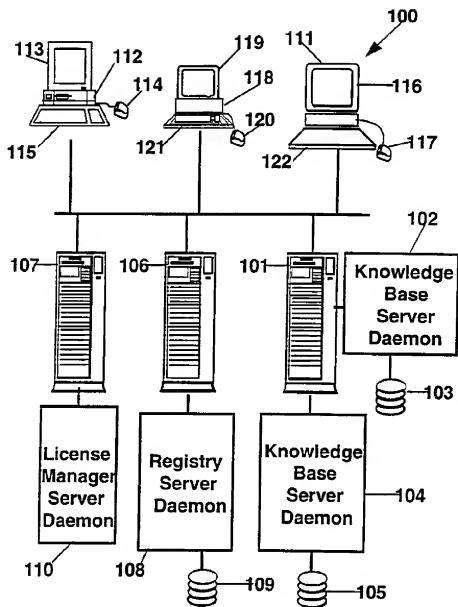
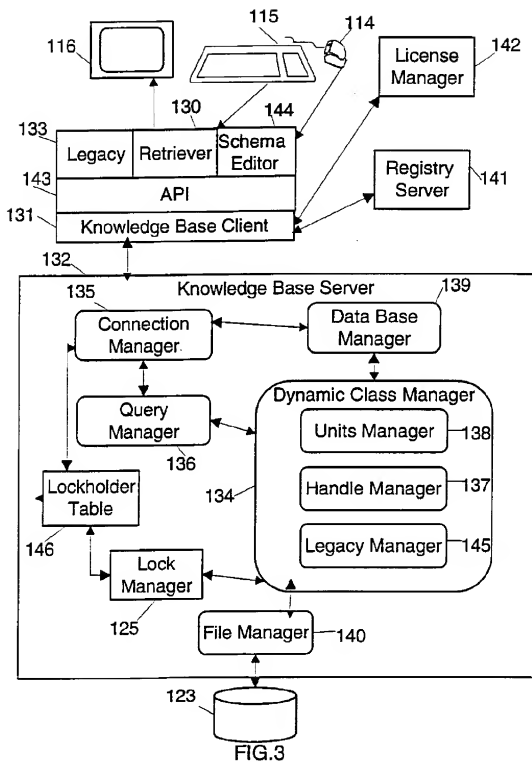


FIG. 2

【图3】



【图 4】

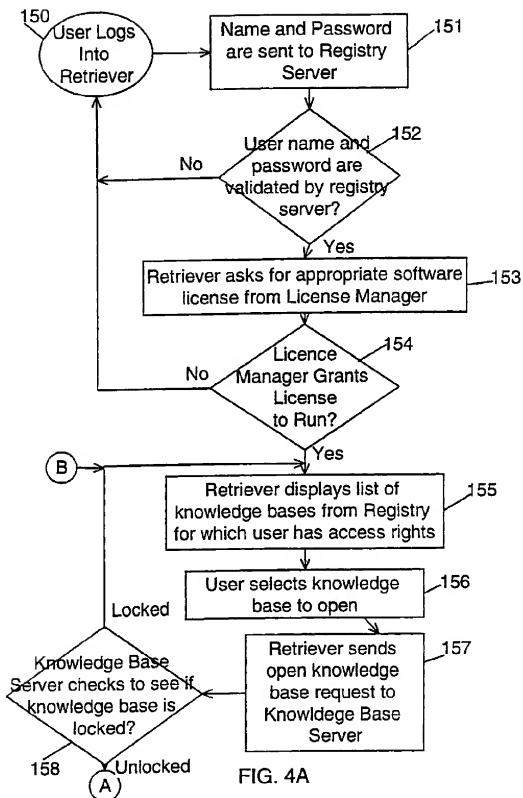


FIG. 4A

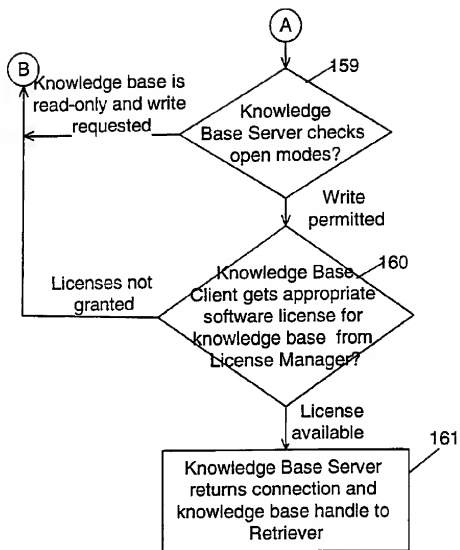


FIG. 4B

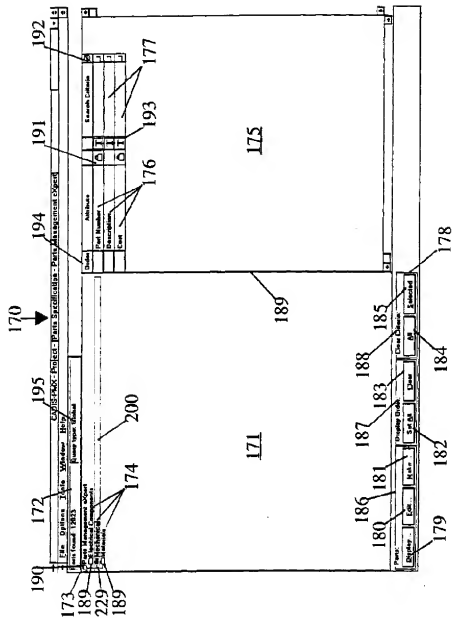


FIG. 5

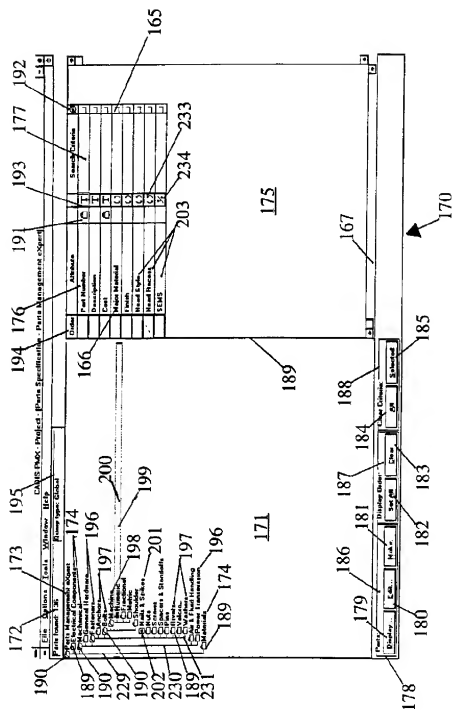


FIG. 6

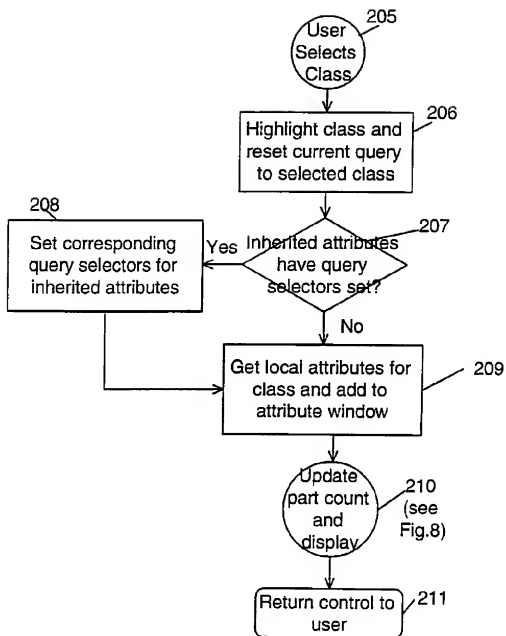


FIG. 7

【图8】

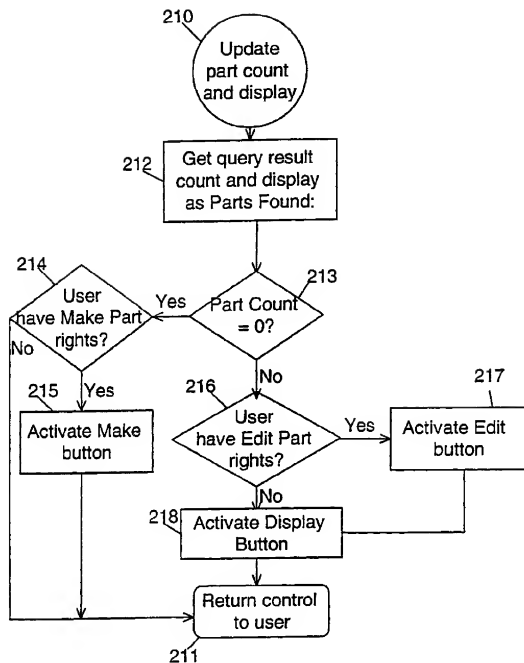


FIG. 8

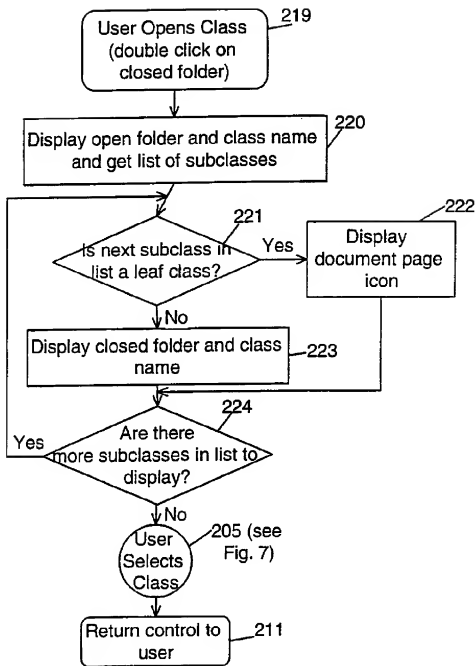


FIG. 9

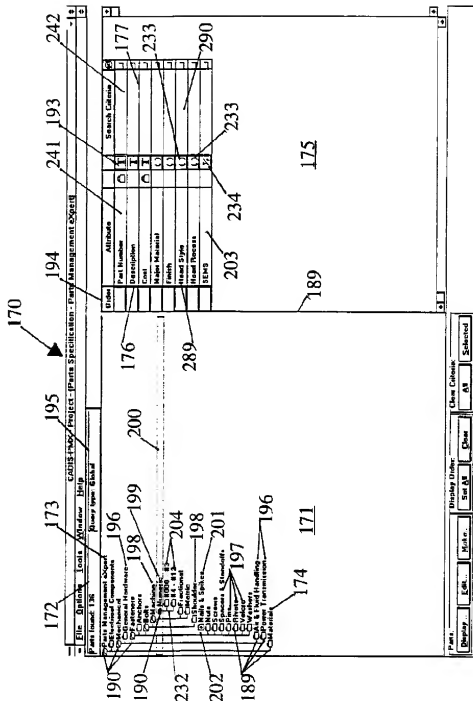


FIG. 10

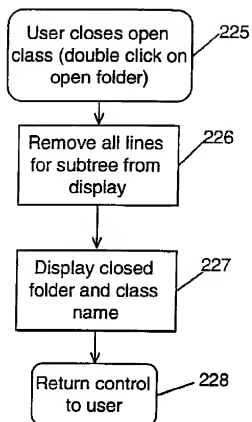


FIG. 11

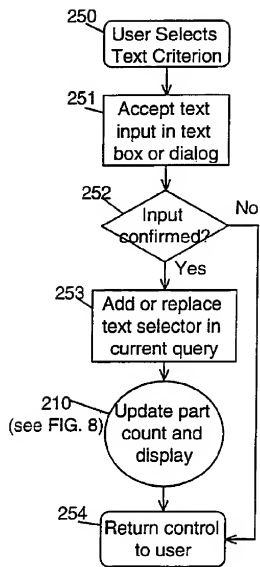


FIG. 12

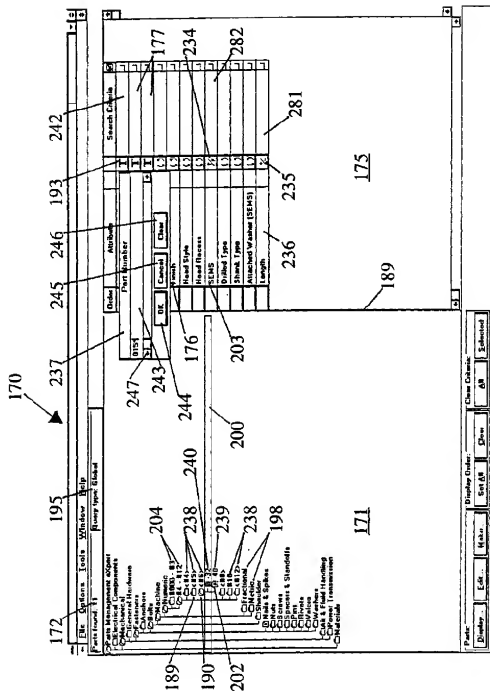


FIG. 13

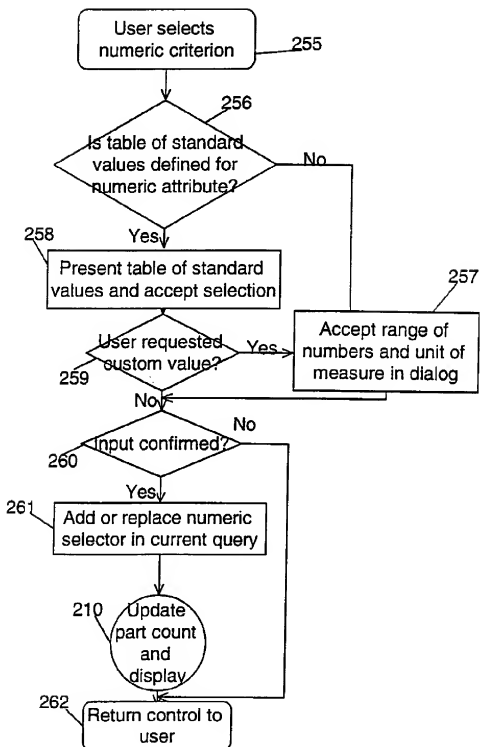


FIG. 14

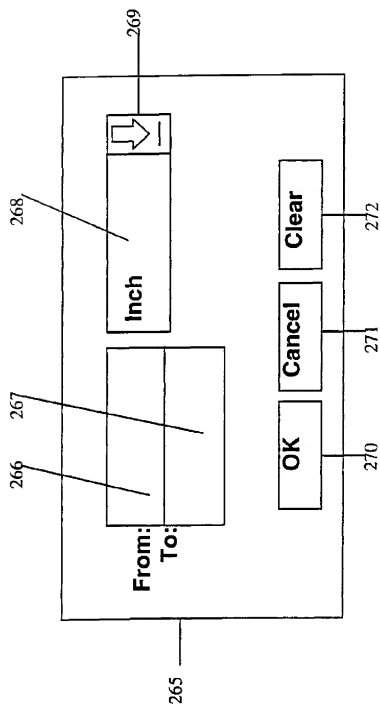


FIG. 15

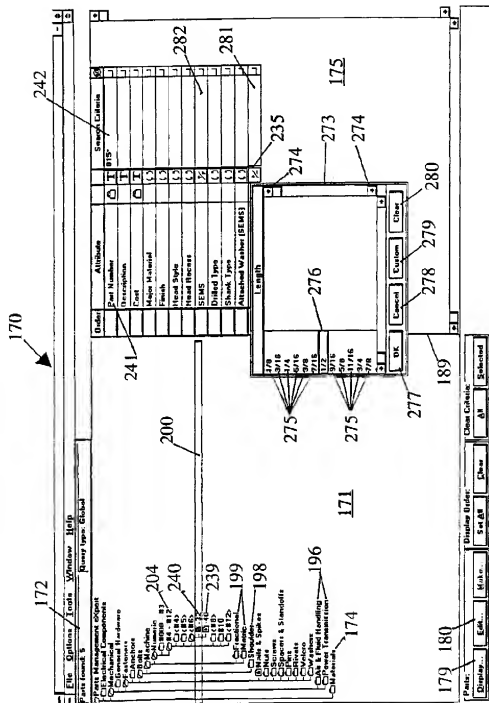


FIG. 16

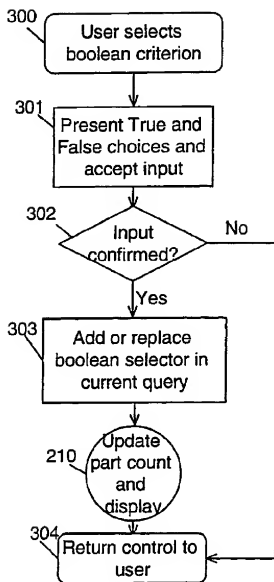


FIG. 17

FIG. 18

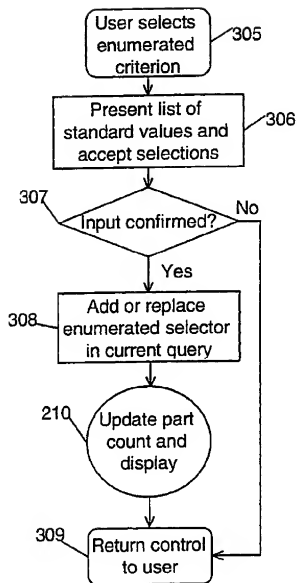
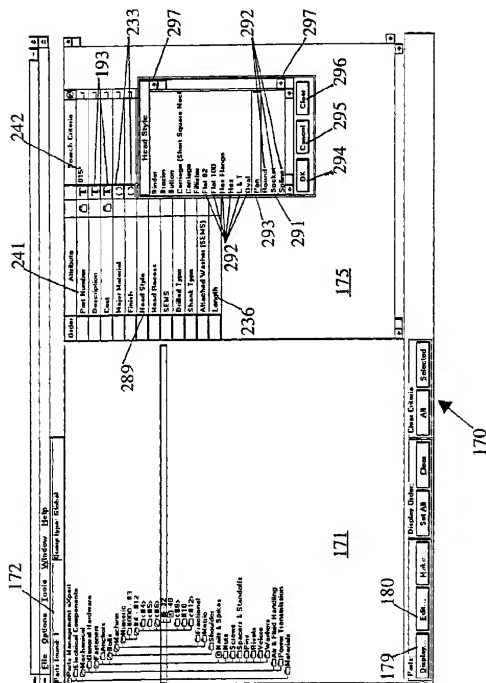


FIG. 19



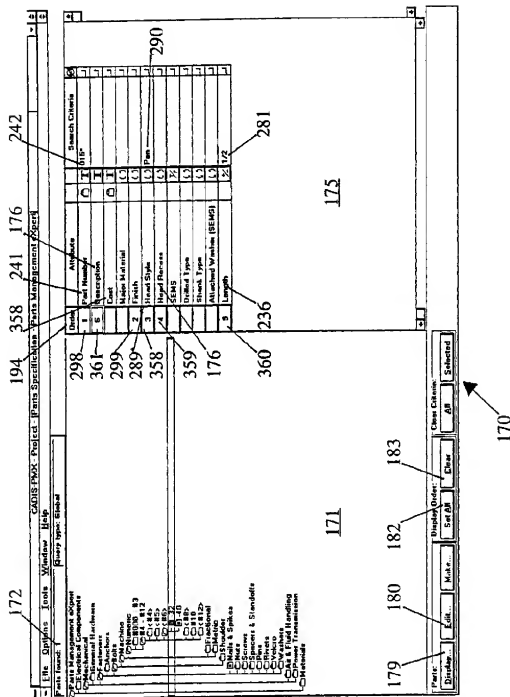


FIG. 21

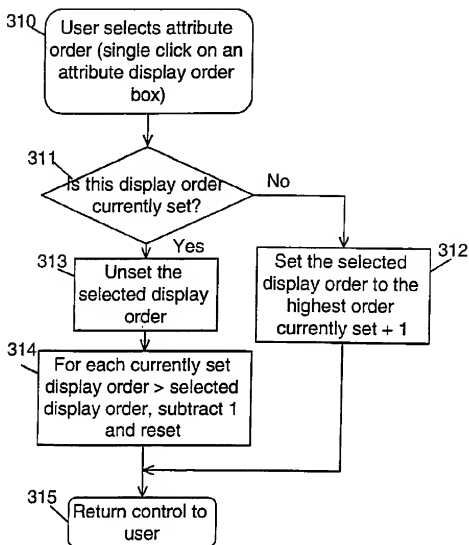


FIG. 22

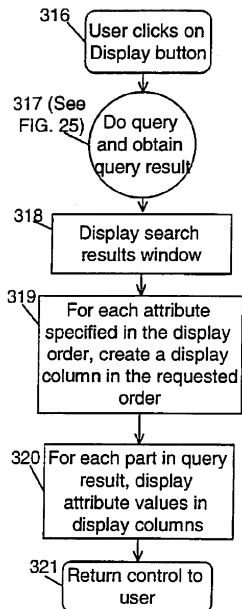


FIG. 23

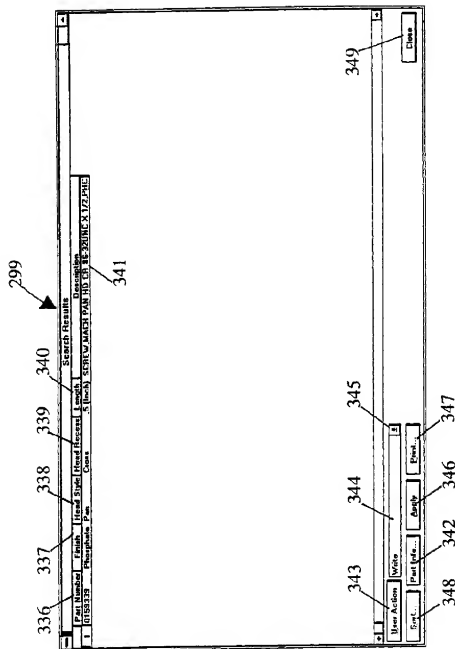


FIG. 24

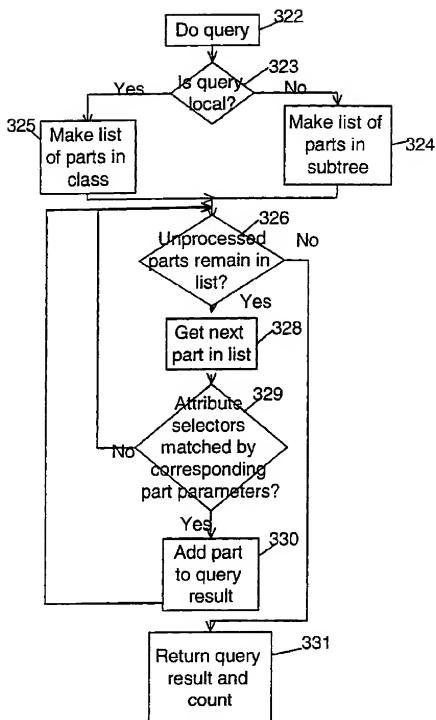


FIG. 25

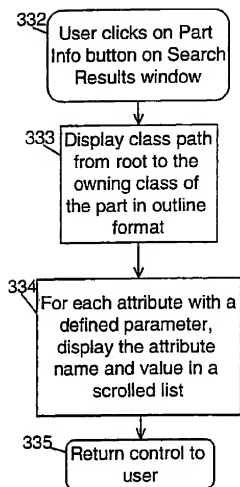


FIG. 26

351

Part Information

350

- Part Management Expert
 - Mechanical
 - Fasteners
 - Bolts
 - Machine
 - Numeric
 - #4 - #12
 - L-#6
 - L-32

353

Attributes

354

Values

Part Number	01593359
Description	SCREW MACH PAN HD CR #6-32UNC X 1/2.PHC
Finish	Phosphate
Head Style	Pan
Head Recess	Cross
Length	.5 Inch

352

OK

356

FIG. 27

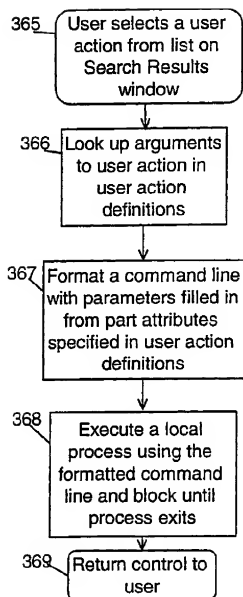


FIG. 28

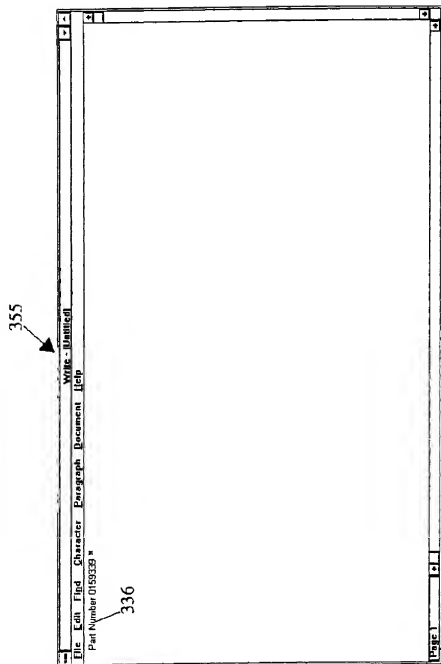


FIG. 29

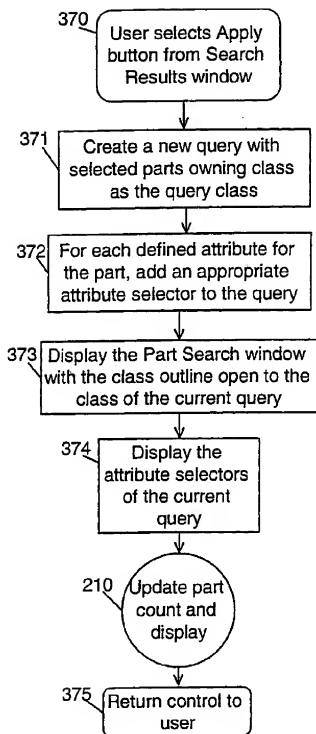


FIG. 30

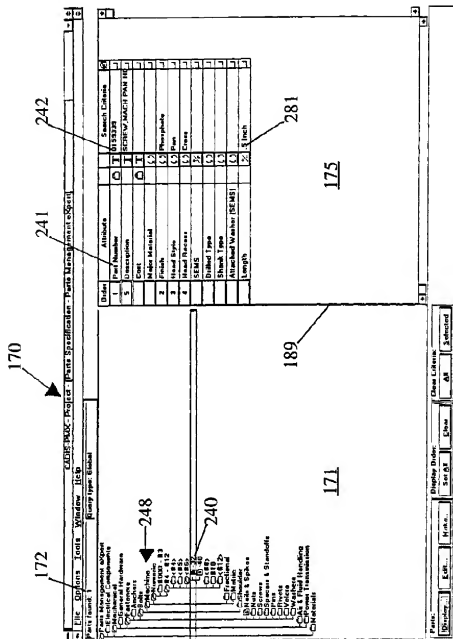


FIG. 31

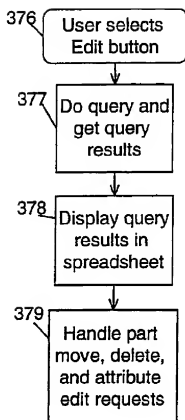


FIG. 32

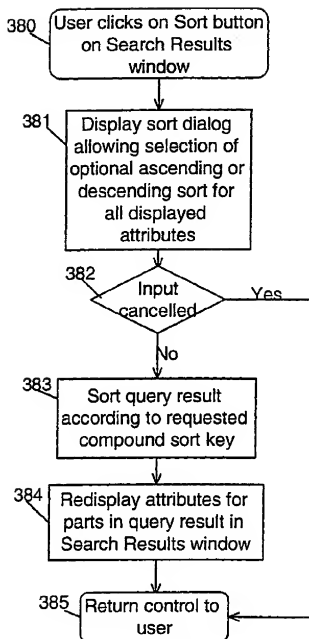


FIG. 33

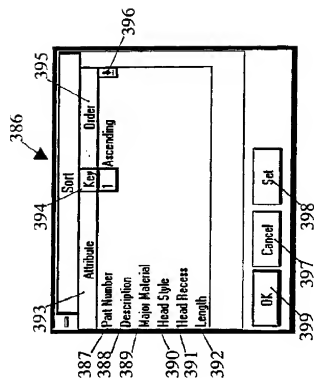


FIG. 34

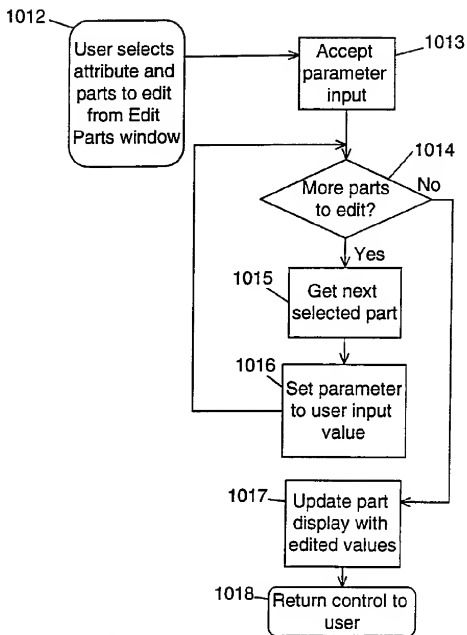


FIG. 35

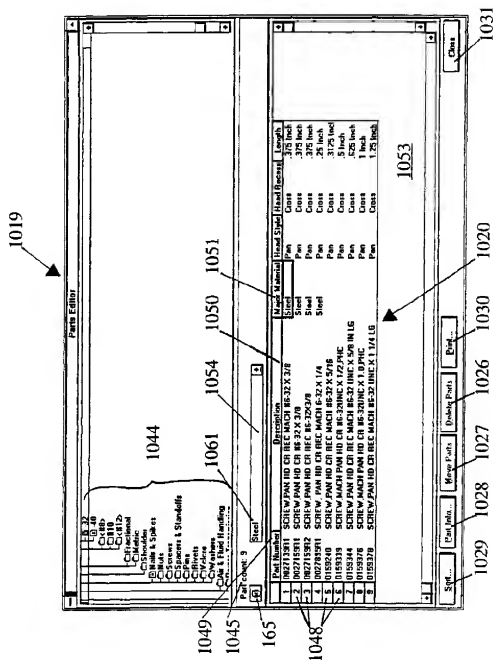
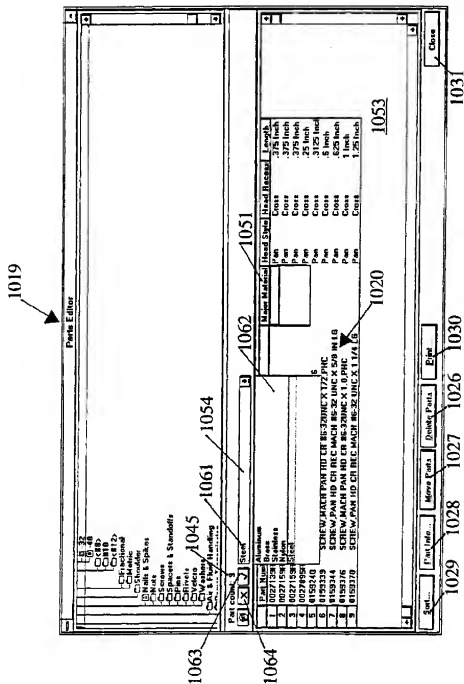


FIG. 36



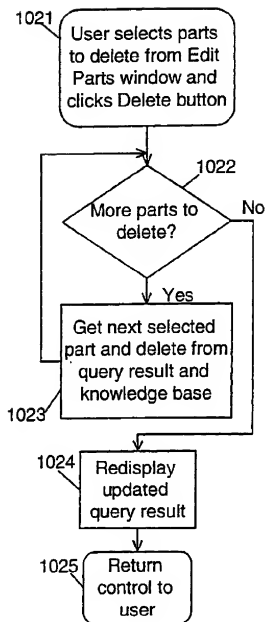


FIG. 38

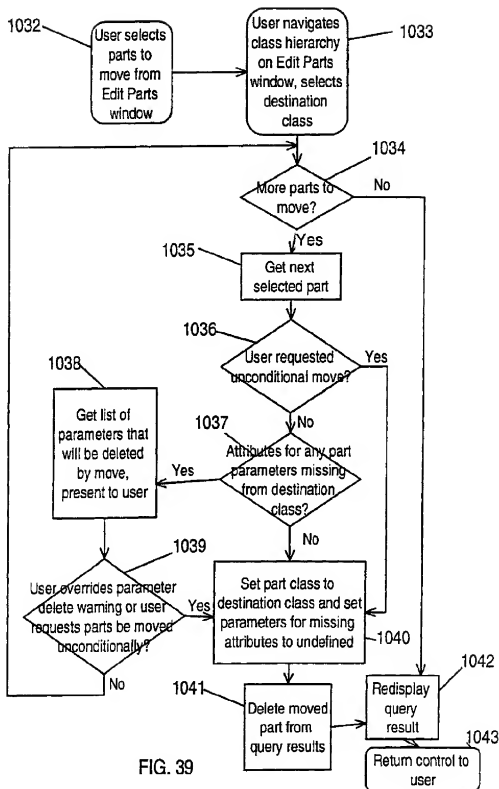


FIG. 39

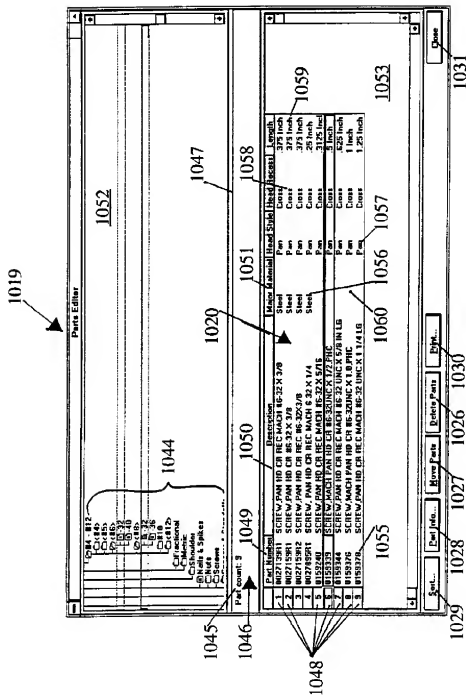


FIG. 40

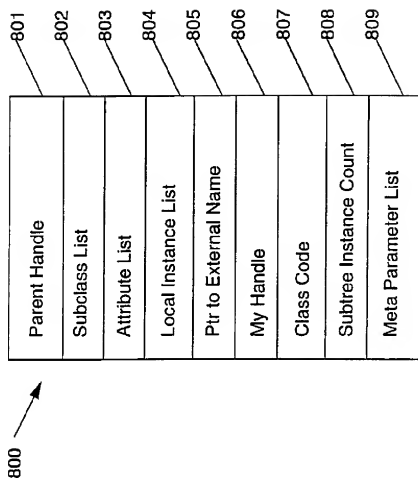


FIG. 41

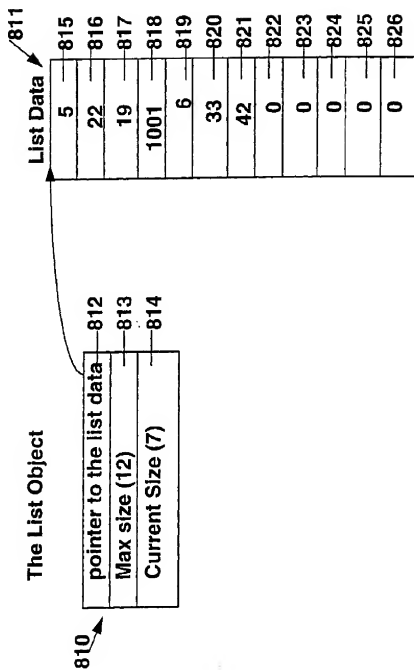


FIG. 42

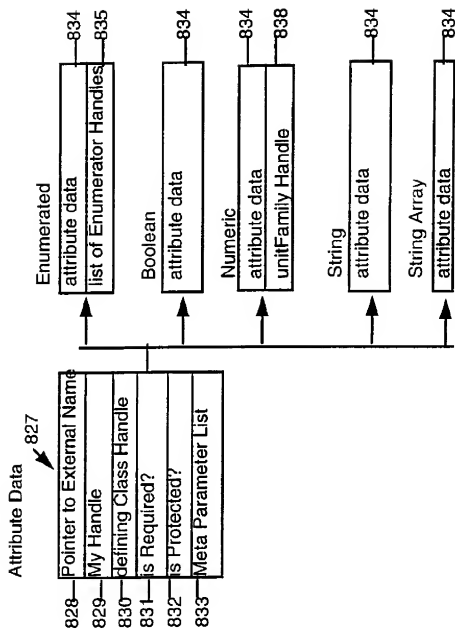


FIG. 43

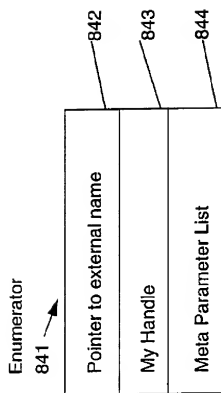


FIG. 44

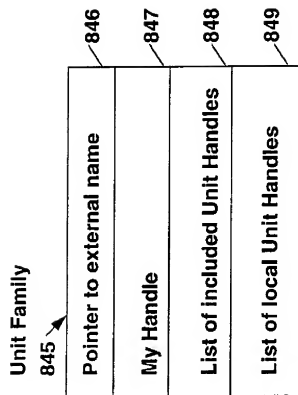


FIG. 45

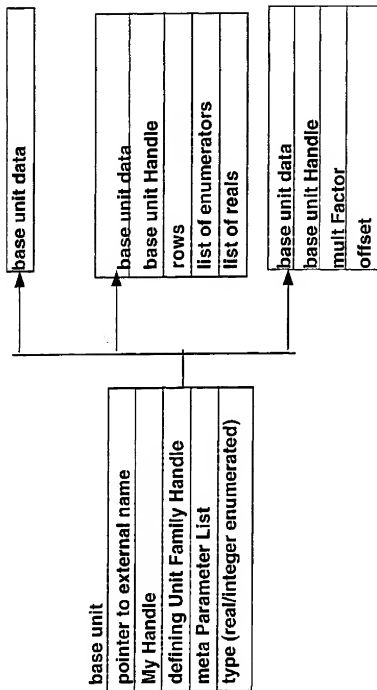


FIG. 46

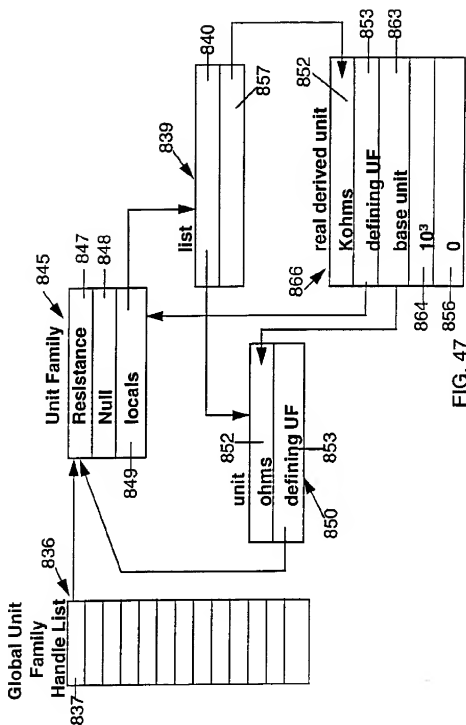


FIG. 47

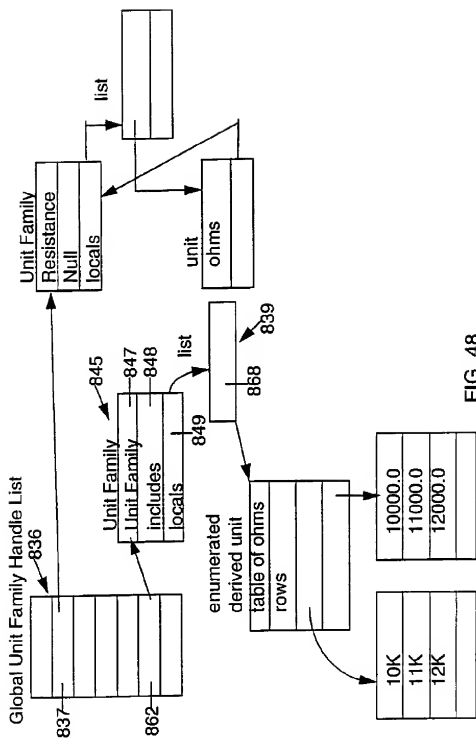


FIG. 48

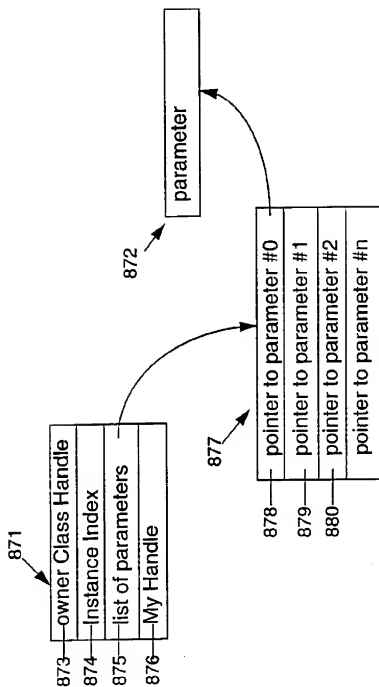


FIG. 49

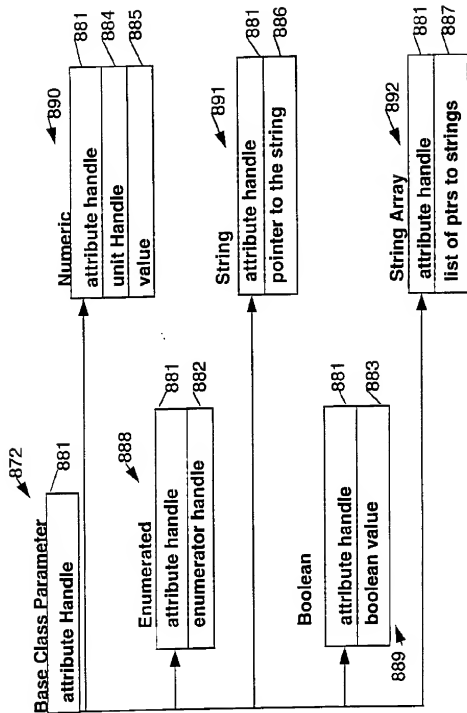


FIG. 50

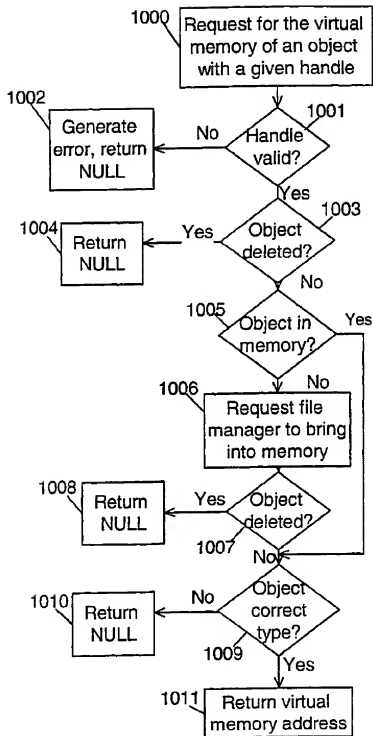


FIG. 52

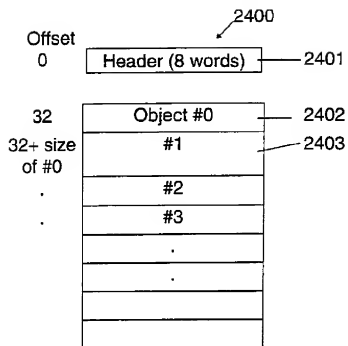


FIG. 53

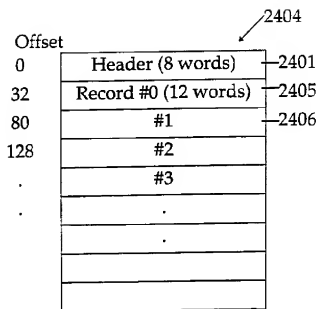


FIG. 54

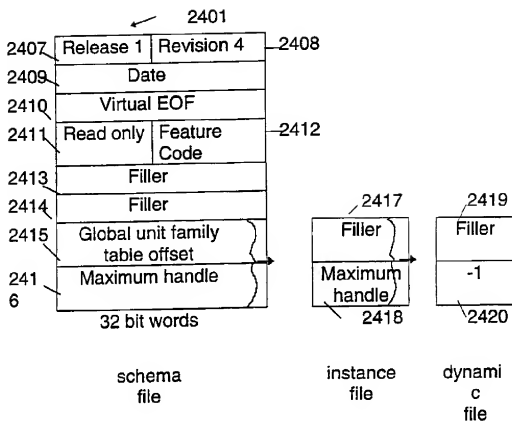


FIG. 55

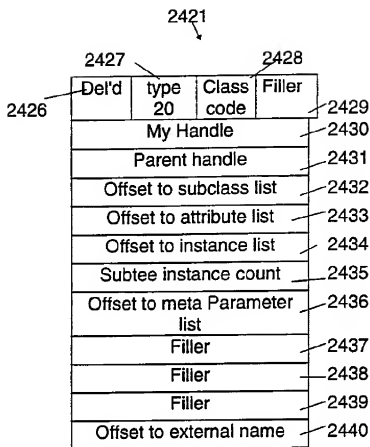


FIG. 56

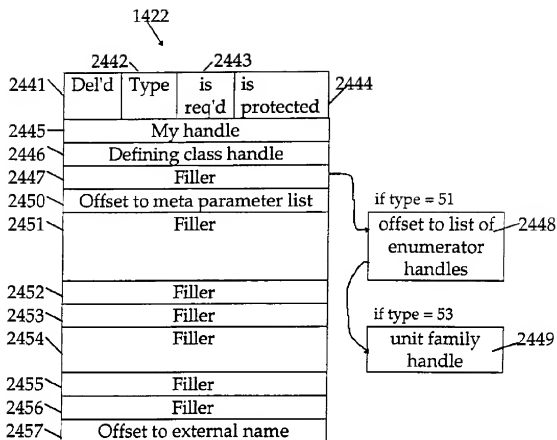


FIG. 57

2423

2458	Del'd	type 60	2459	Filler	2460
	My Handle				2461
	Offset to meta parameter list				2462
	Filler				2463
	Filler				2464
	Filler				2465
	Filler				2466
	Filler				2467
	Filler				2468
	Filler				2469
	Filler				2470
	Offset to external name				2471

FIG. 58

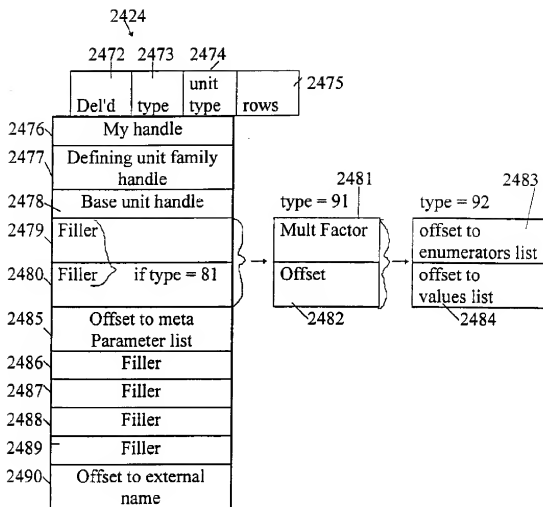


FIG. 59

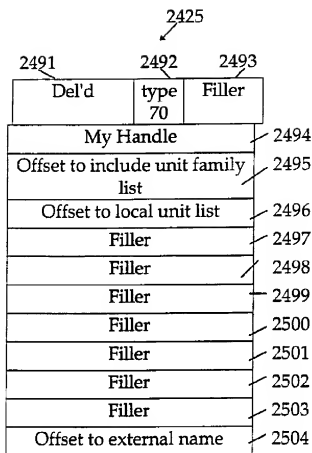


FIG. 60

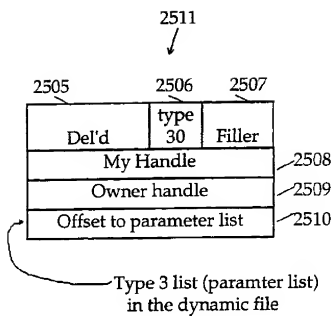


FIG. 61

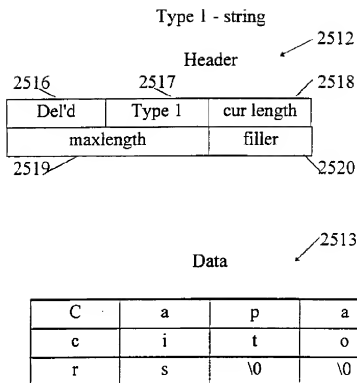


FIG. 62

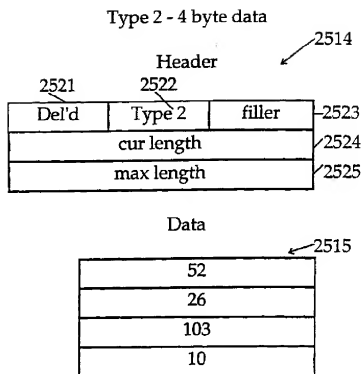


FIG. 63

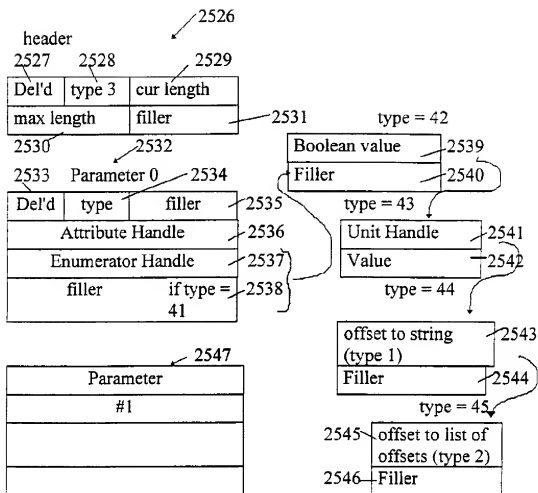


FIG. 64

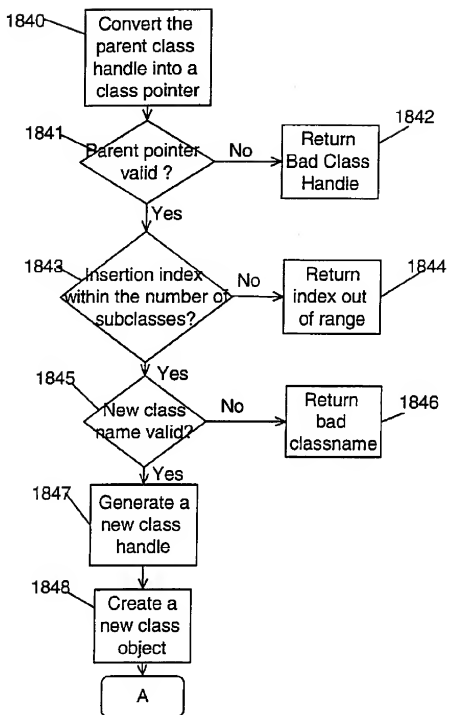


FIG. 65

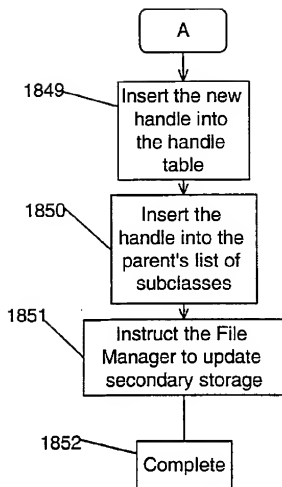


FIG. 66

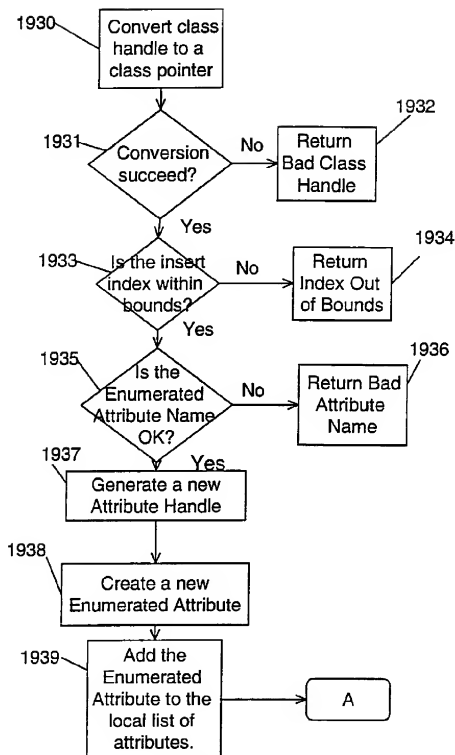


FIG. 67

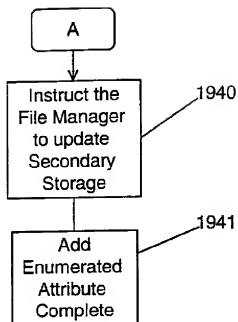


FIG. 68

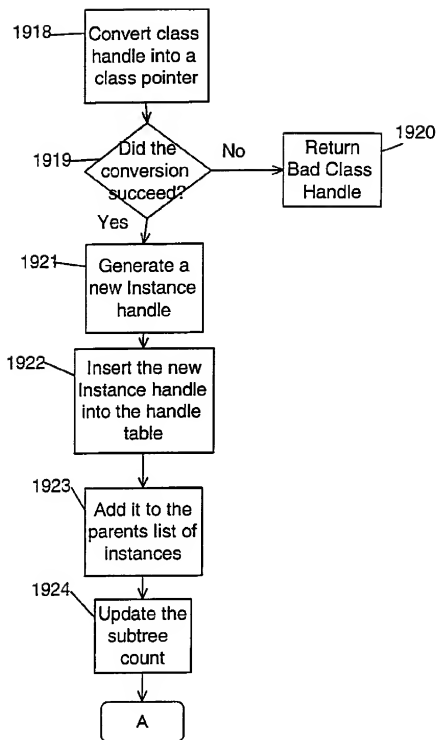


FIG. 69

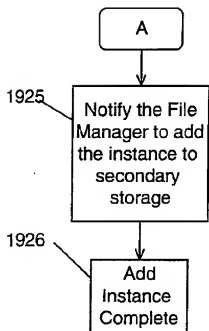


FIG. 70

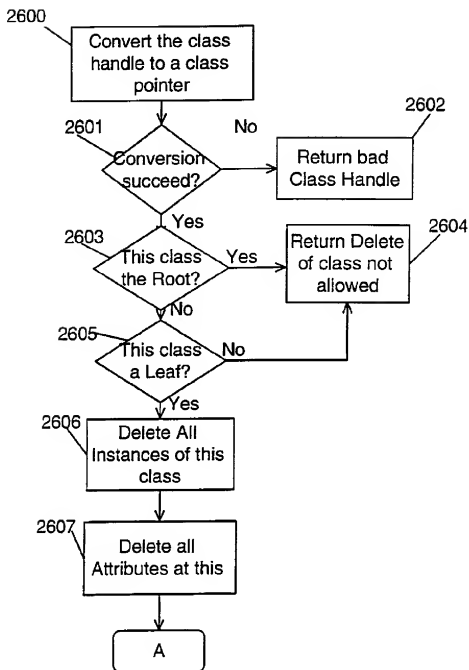


FIG. 71

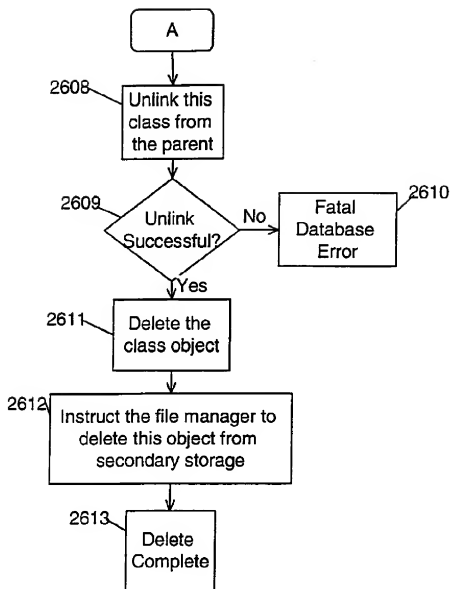


FIG. 72

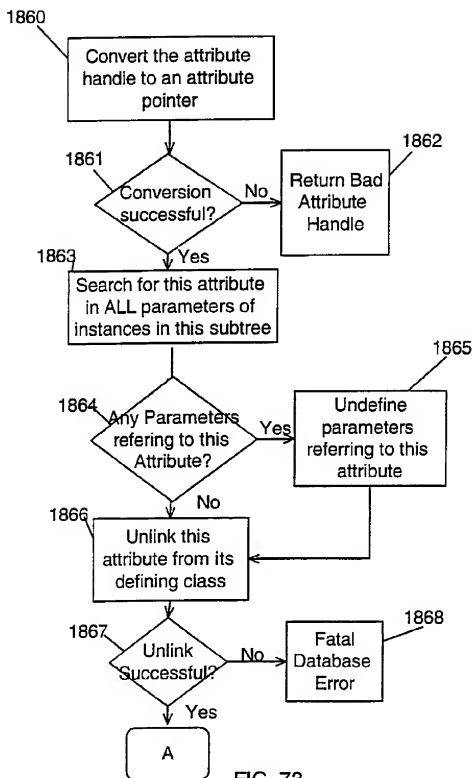


FIG. 73

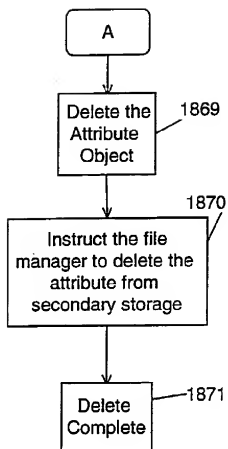


FIG. 74

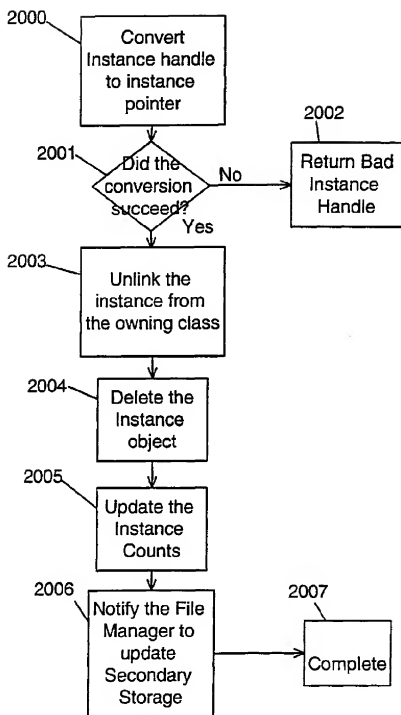


FIG. 75

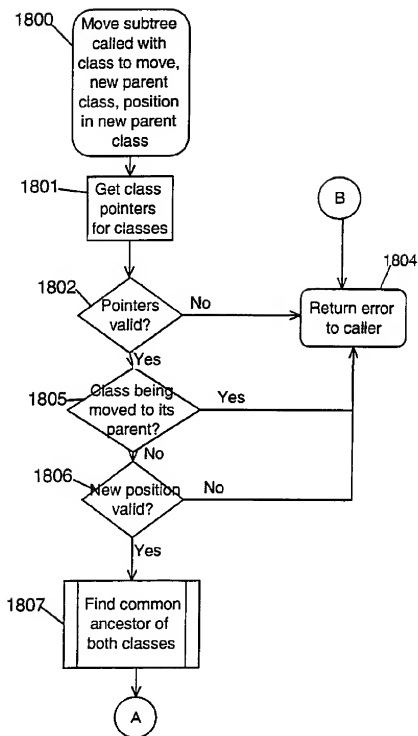


FIG. 76

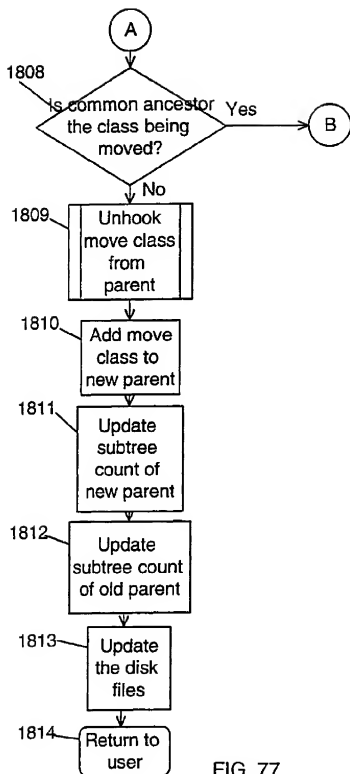


FIG. 77

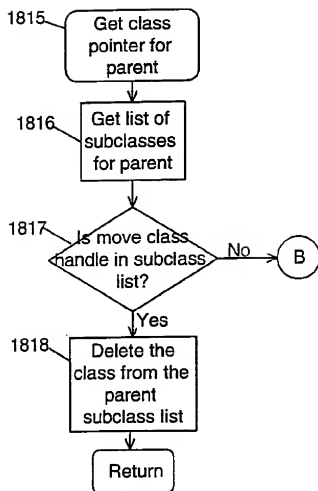


FIG. 78

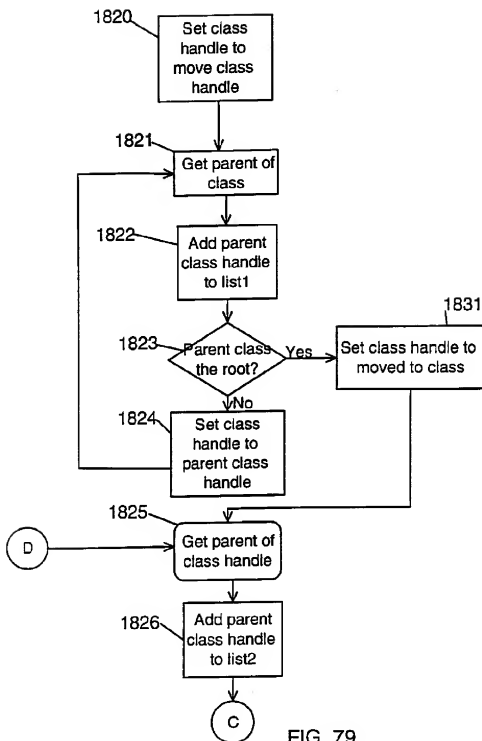


FIG. 79

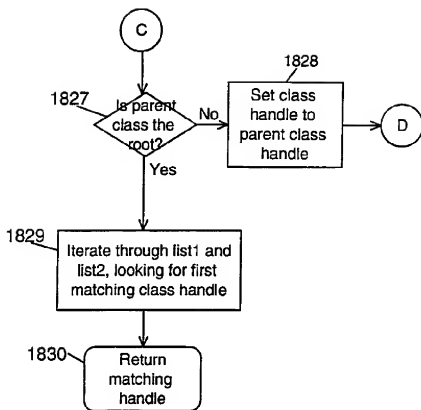


FIG. 80

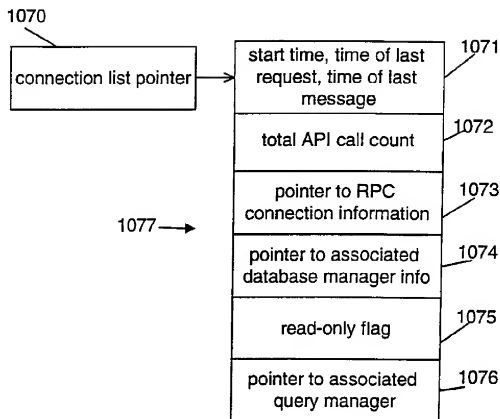
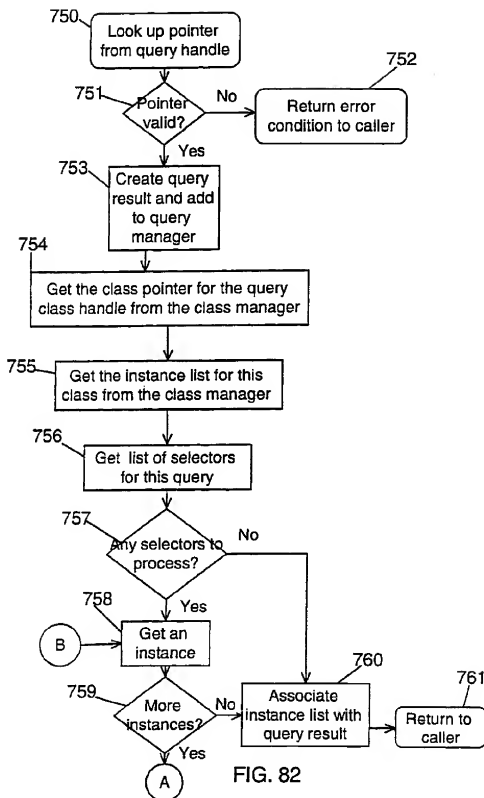


FIG. 81



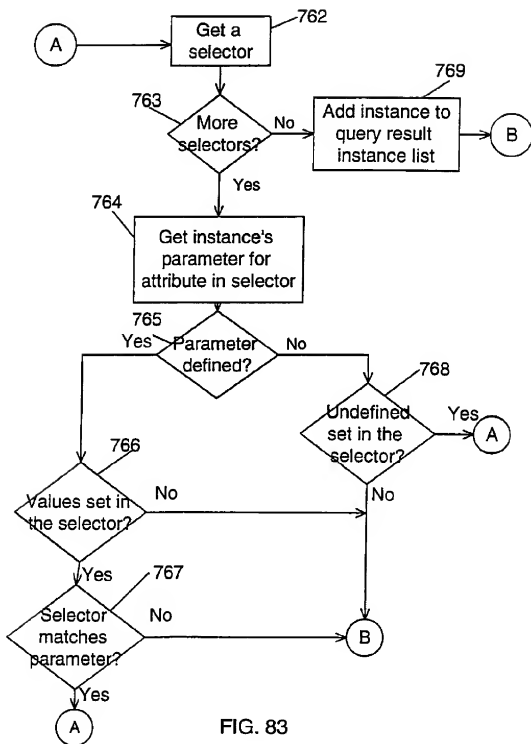


FIG. 83

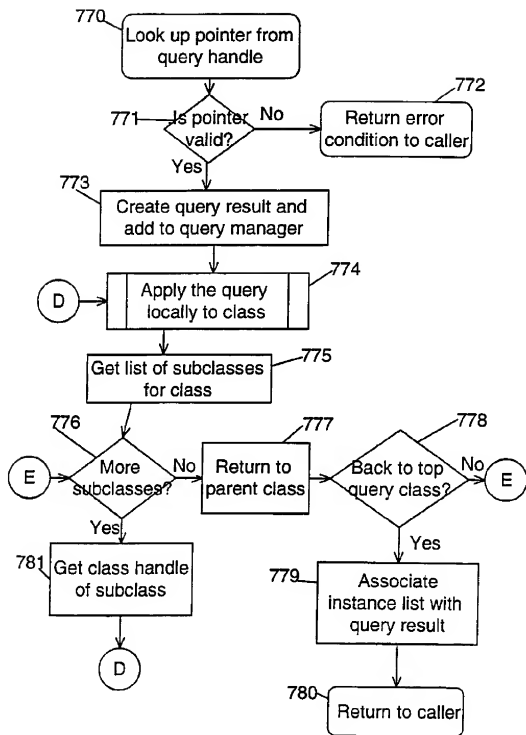


FIG. 84

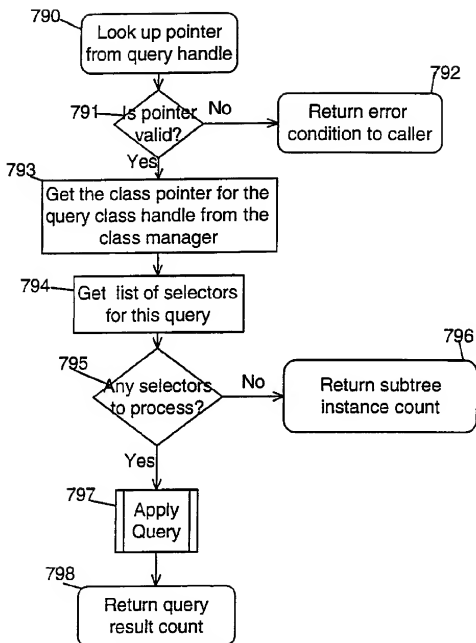


FIG. 85

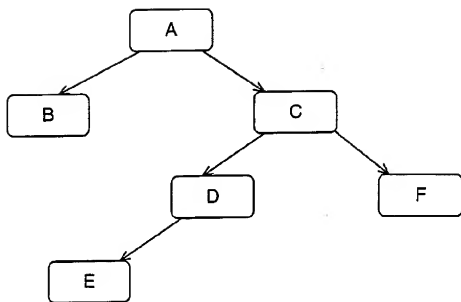


FIG. 86

Match Component	Component	Matched?
Base Number	2901	
Prefix	LM	No
Suffix	B	Yes
Manufacturer	AMD	No
# of Classes Found	1	Yes

FIG. 87

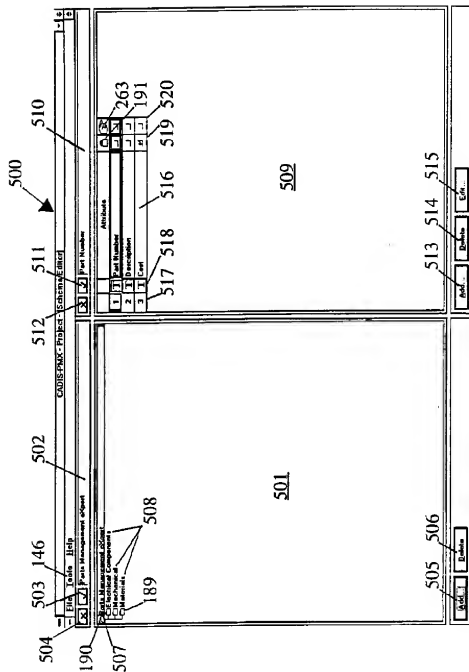


FIG. 89

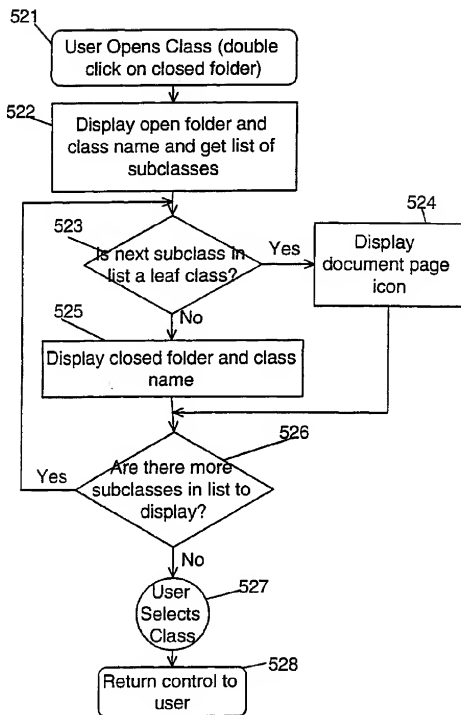


FIG. 90

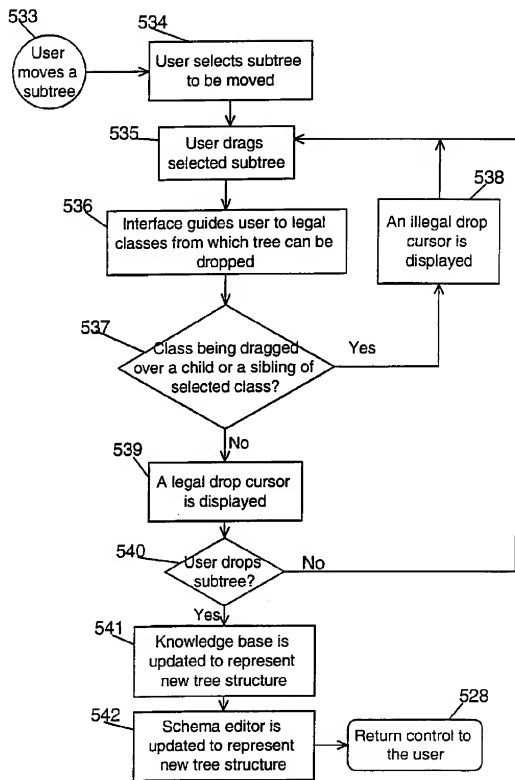
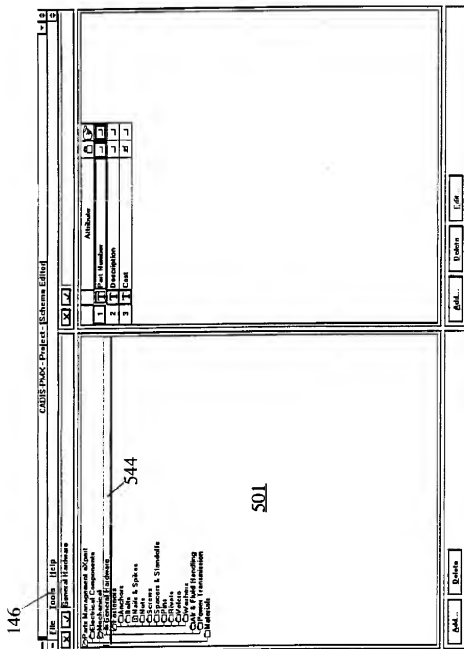


FIG. 92



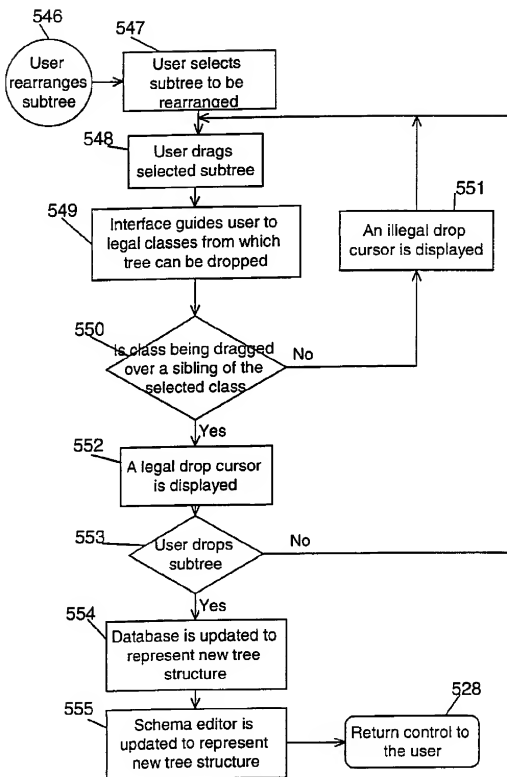


FIG. 95

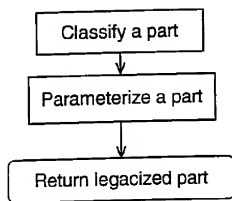


FIG. 96

File Edit Help ZASUS MAX Project Schema Editor 1 2 3

☒ Forward Hardware ☒ ☐ ☐

Project Management Objects
 Object-oriented
 Components
 C-Modules
 Signals
 Signals & States
 States
 C-Modules
 C-Modules & Standards
 C-Modules
 Divisions
 Data & Flight Handling
 C-Modules
 C-Modules

557

	Unit Number	Attribute
1	1	0
2	1	1
3	1	1
4	1	1

FIG. 97

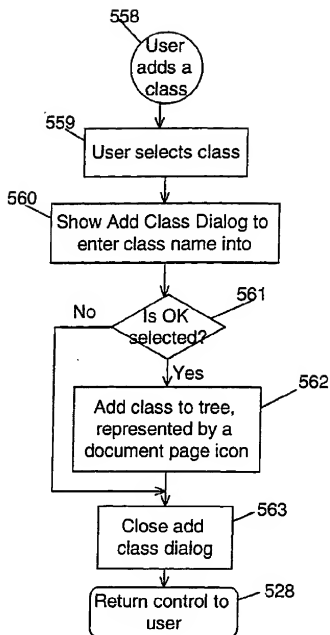


FIG. 98

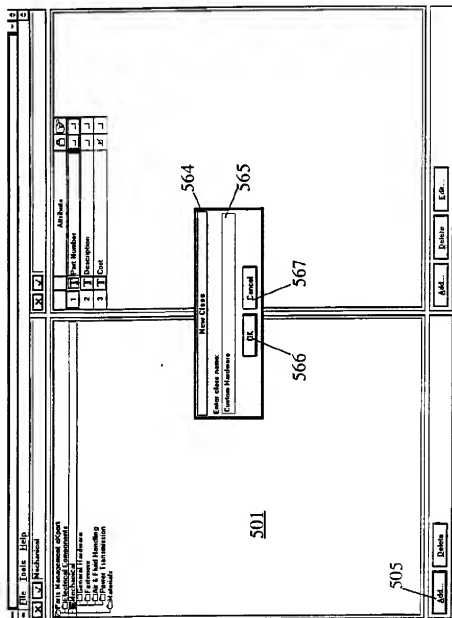
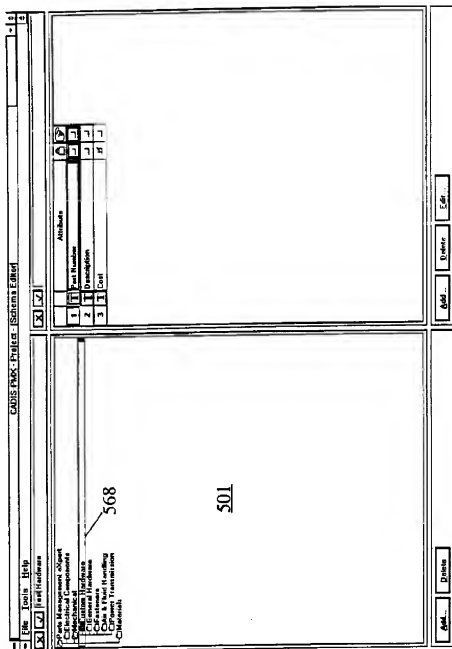


FIG. 99



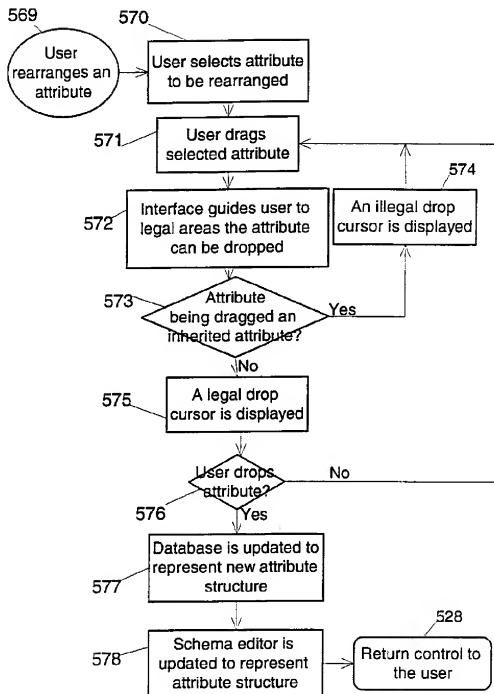


FIG. 101

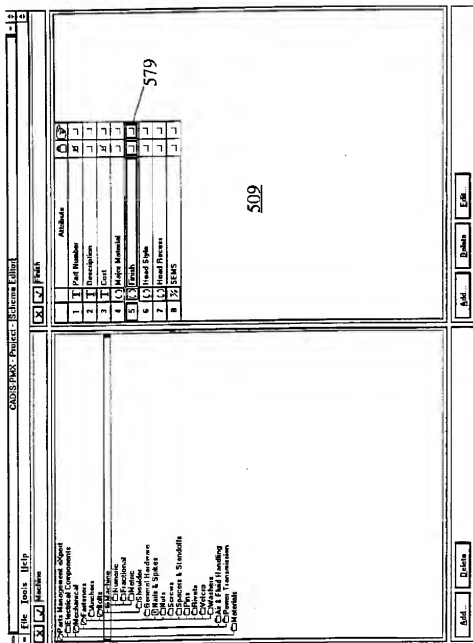


FIG. 102

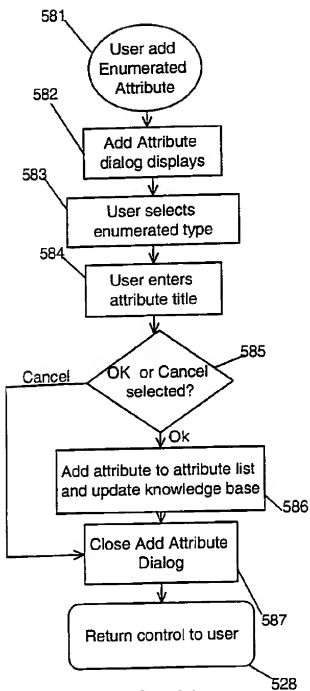


FIG. 104

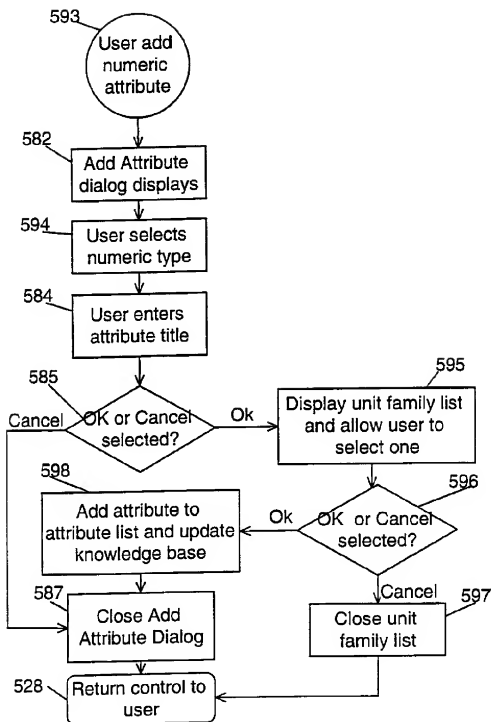
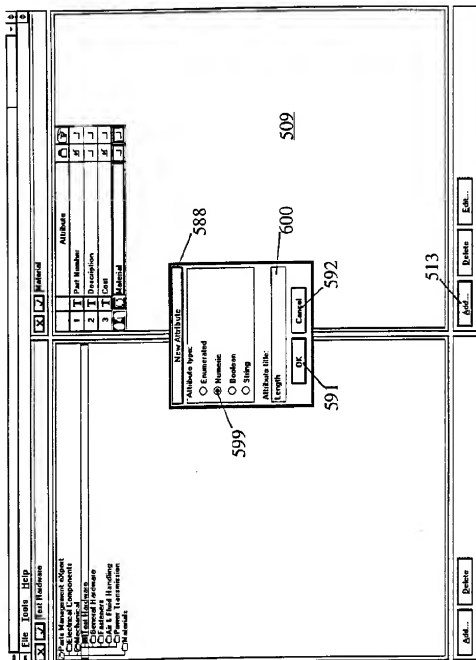
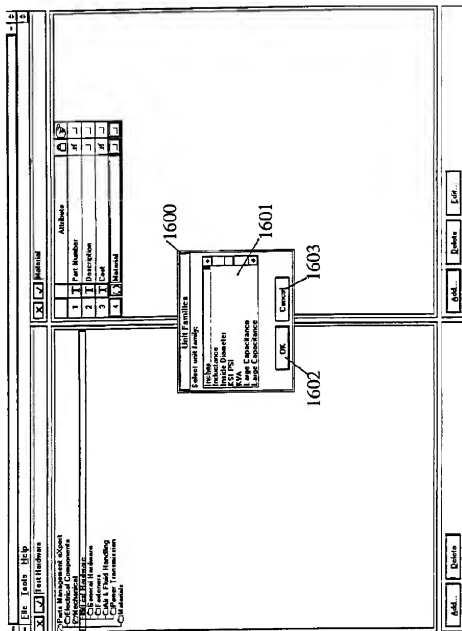


FIG. 106





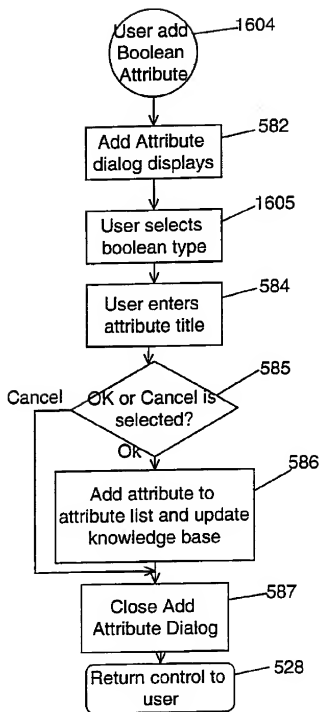


FIG. 109

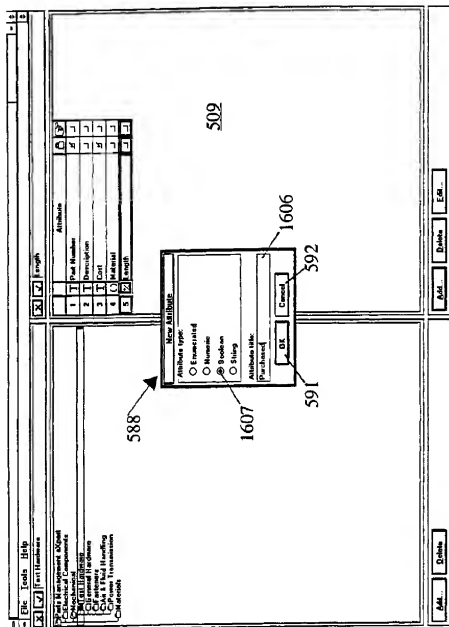


FIG. 110

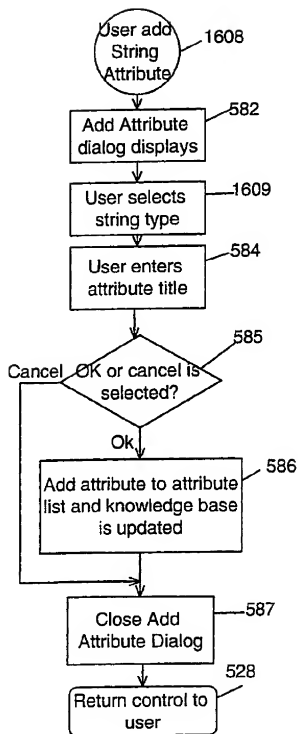


FIG. 111

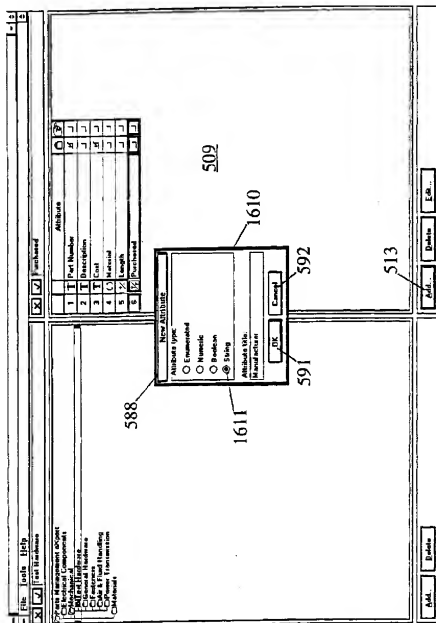


FIG. 112

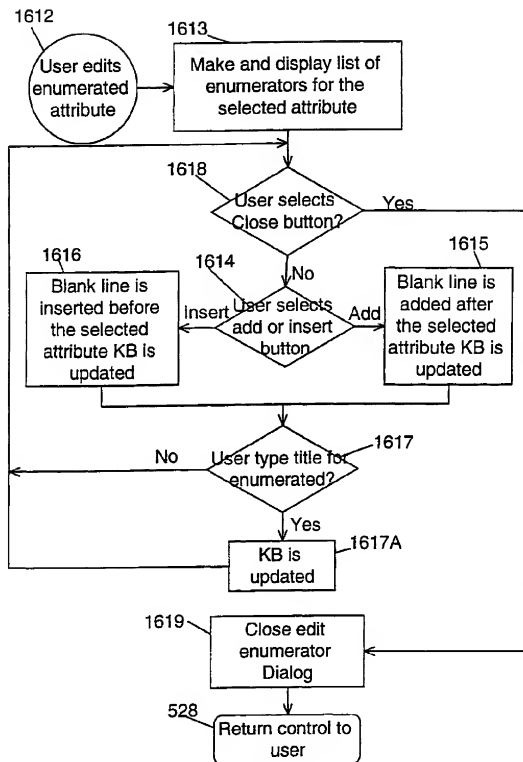


FIG. 113

CADIS-PMX - Project - [Schema Editor]

File Tools Help

☒ Test Hardware

- ☒ System Management - SCOUT
- ☒ Mechanical Components
- ☒ Mechanical
- ☒ Test Hardware
- ☒ General Hardware
- ☒ Cables
- ☒ Cold & Fluid Handling
- ☒ Power Transmission
- ☒ Materials

☒ Material

	Altitude	Altitude
1 Part Number		1
2 Description		1
3 Cost		1
4 Material		1
5 Length		1
6 Purchased		1
7 Manufacturer		1

1620

Aluminum

Edit Enumerator

1623

1621

509

515

Add...

Delete

Add...

Delete

Edit...

FIG. 114

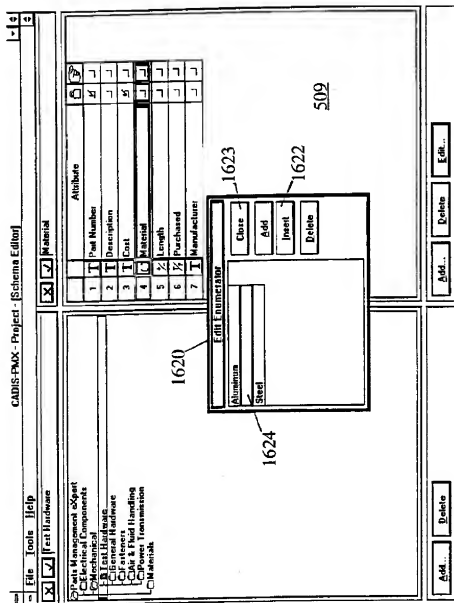


FIG. 115

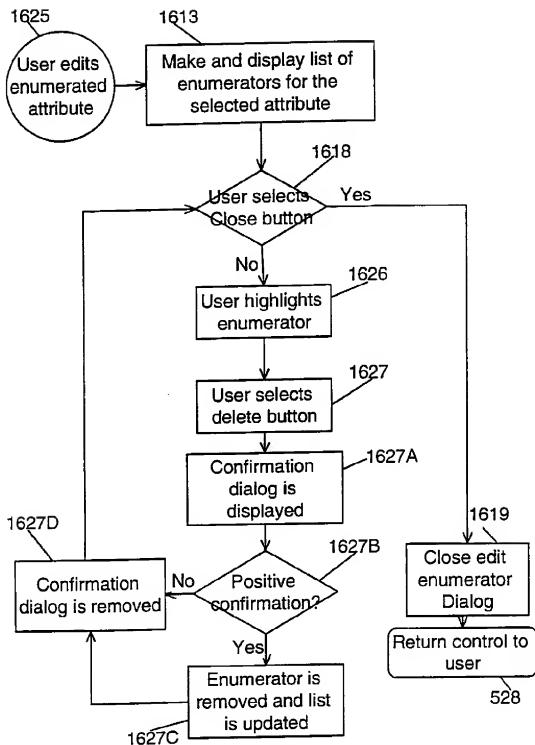


FIG. 116

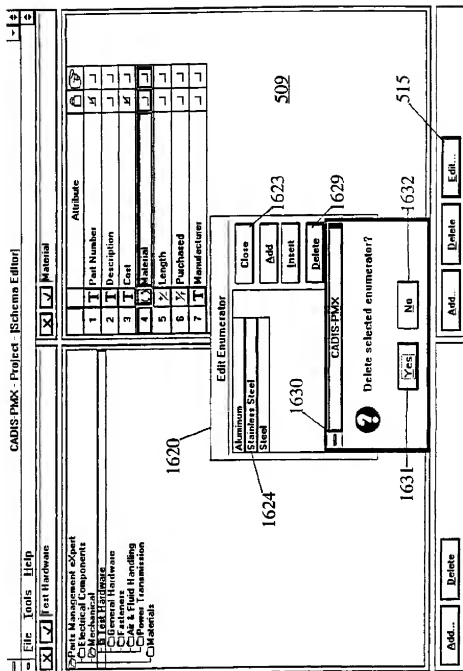


FIG. 117

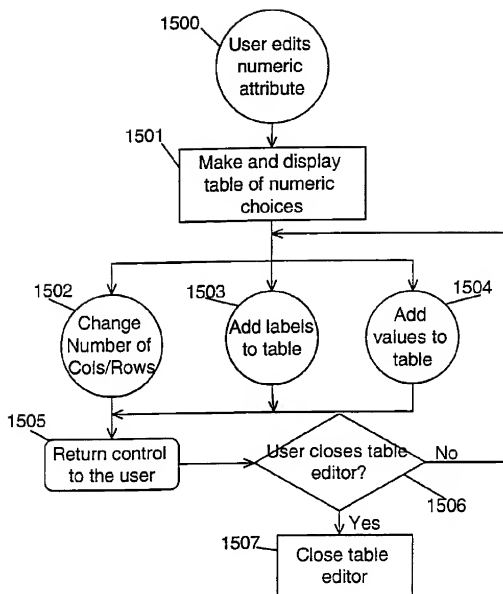


FIG. 118

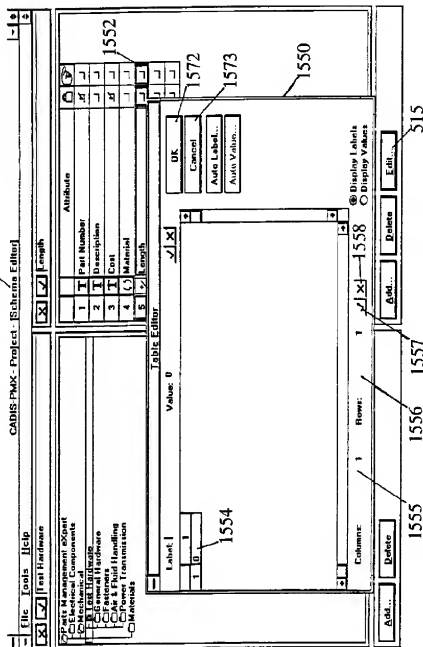


FIG. 119

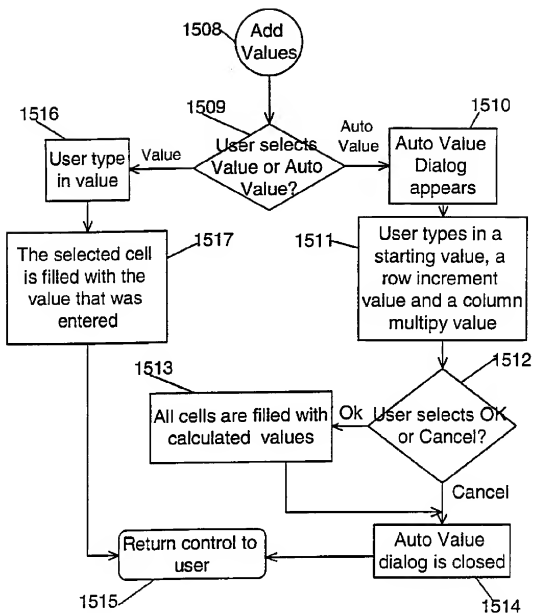


FIG. 120

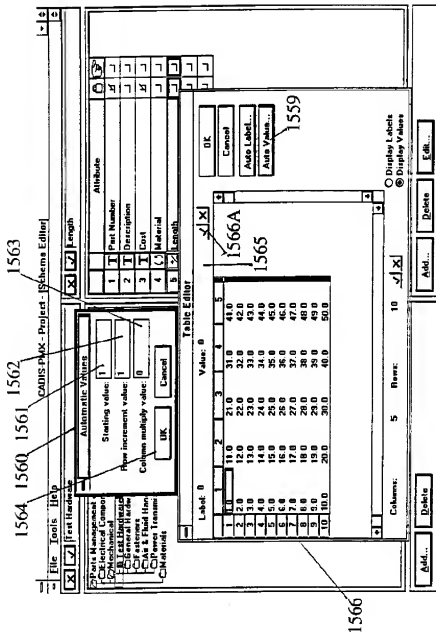


FIG. 121

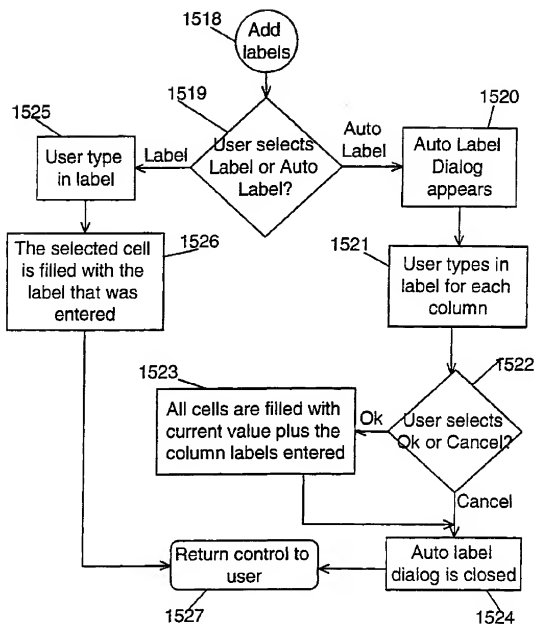


FIG. 122

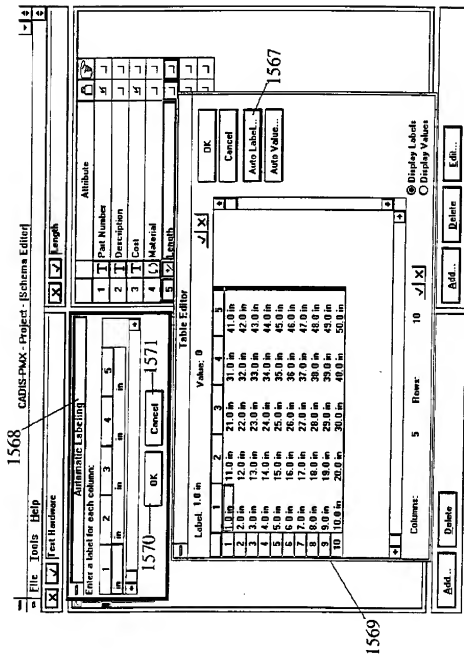


FIG. 123

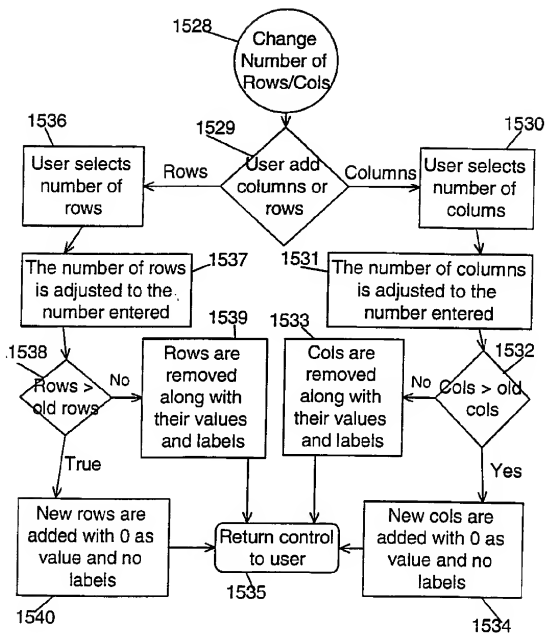


FIG. 124

```
import [-u user] [-p password] [-d dbname] [-P]
[-U] [-X] [-M] [-v] [-r] import_file
```

-u	use user name 'username' when doing login, if none given login id is used
-p	use specified password with login,
-d	the logical database name to connect to
-P	turn on progress statistics
-U	don't import a parameter unless the instance is unique with respect to the search keys
-X	don't import a parameter unless there are no matching instances. Must be used in conjunction with -M.
-M	if a match is not found, make a new instance
-v	turn on verbose mode
-r	remove the matched instances from the knowledge base
import_file	name of the file containing the import info.

FIG. 125

```
simp [-u user] [ -p password ] -d kdbname -o outfile
[-f] [-P] [-U] [-v] [-n] import_file import_map
```

-u	use user name 'user name' when doing login, if none given login id is used
-p	use specified password with login
-d	the logical database name to connect to
-f	the field to match on.
-P	turn on progress statistics
-v	turn on verbose mode
-o	output file for instances not imported
-n	no mapfile - use defining class in import file as import class
import_file	name of the file containing the import info.
import_map	name of the file containing the map information

FIG. 126

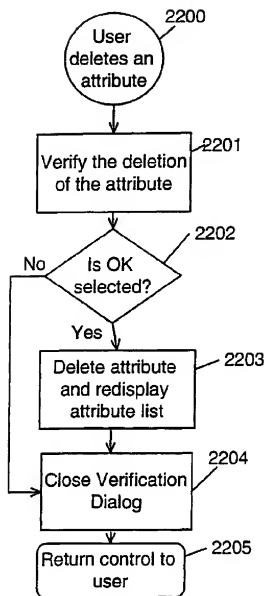


FIG. 127

File Tools Help

CADIS PUX - Project - Schema Editor

Test Hardware

Project Management eXport
 Technical Components
 Test Hardware
 General Hardware
 As & Fluid Handling
 Power Transmission
 Materials

500

2207

2210 2209 2208

509

519

		Length	Attribute
1	T	Part Number	A
2	T	Description	A
3	T	Cost	A
4	T	Material	A
5	T	Length	A
6	T	Manufacturer	A

Add...

Delete

Add...

Delete

Edit...

FIG. 129

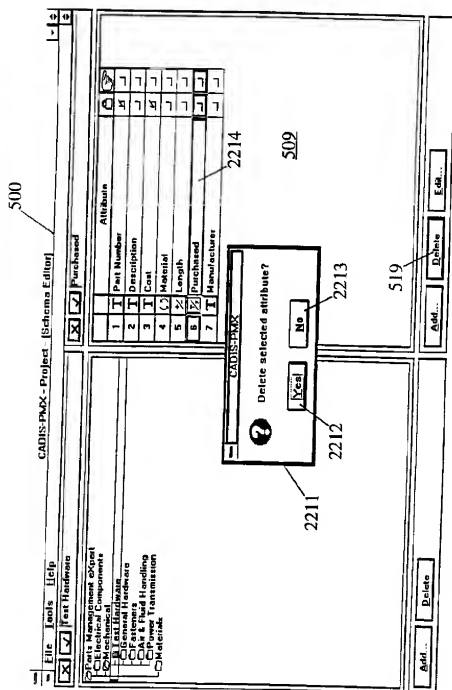
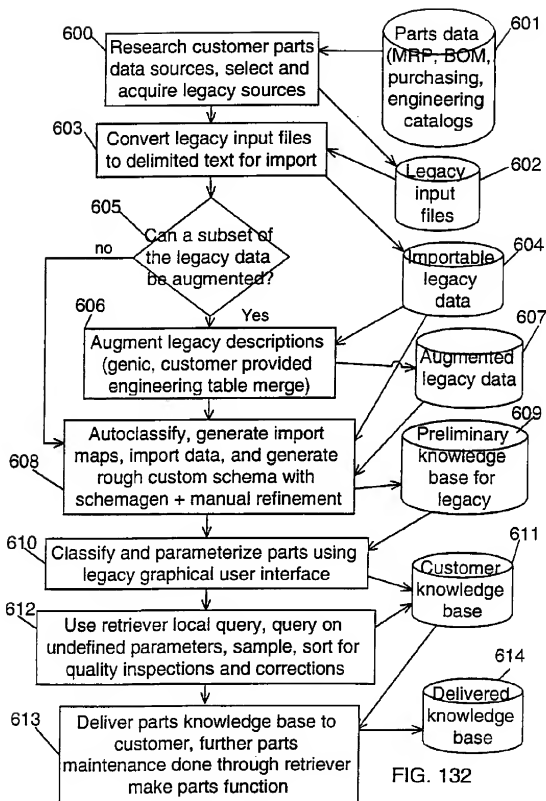


FIG. 130

Match Component	Component	Matched?
Base Number	2901	
Prefix		Yes
Suffix	A	No
Manufacturer	Intel	Yes
# of Classes Found	1	Yes

FIG. 131



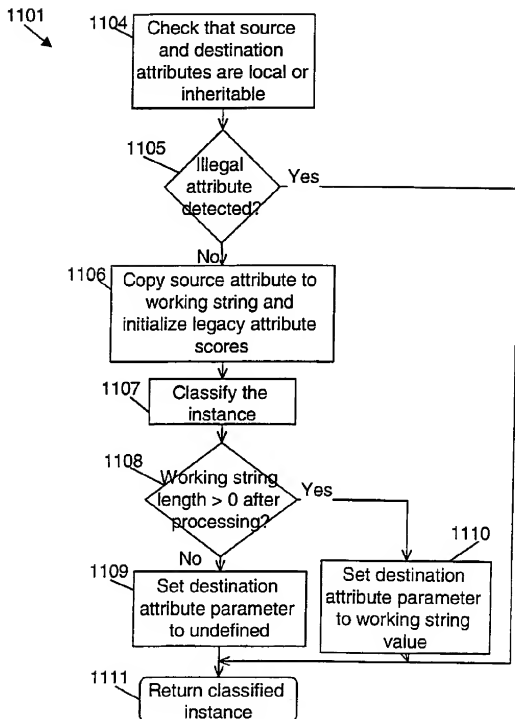


FIG. 133

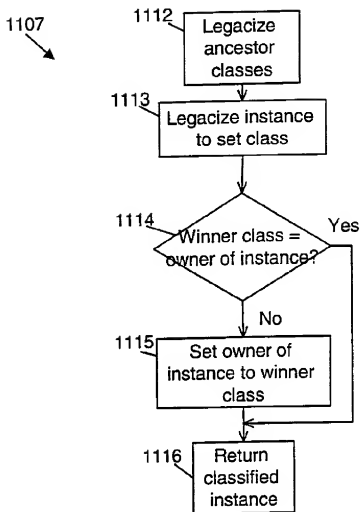


FIG. 134

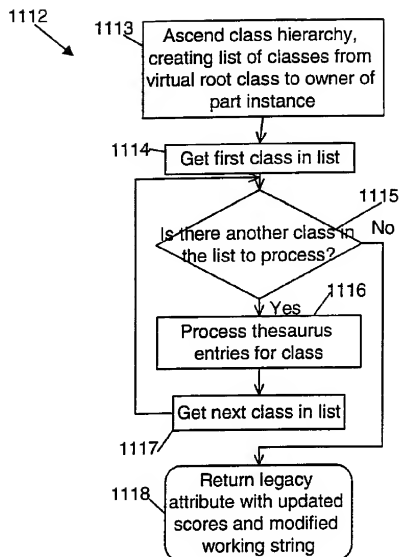


FIG.135

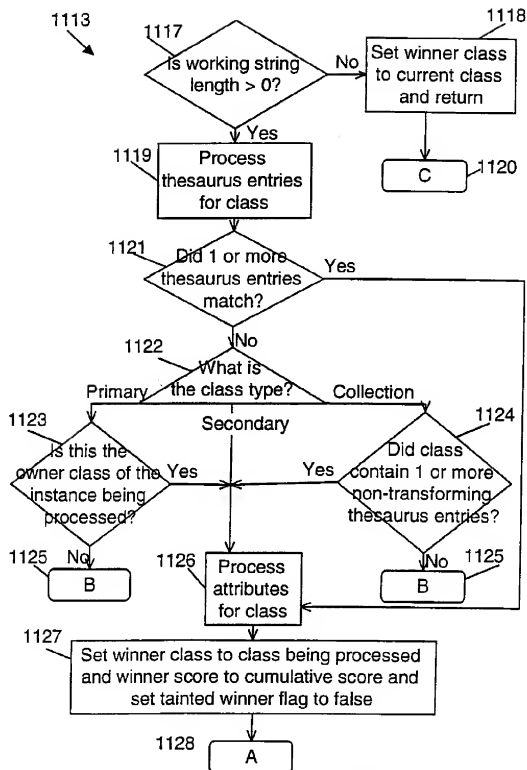


FIG. 136

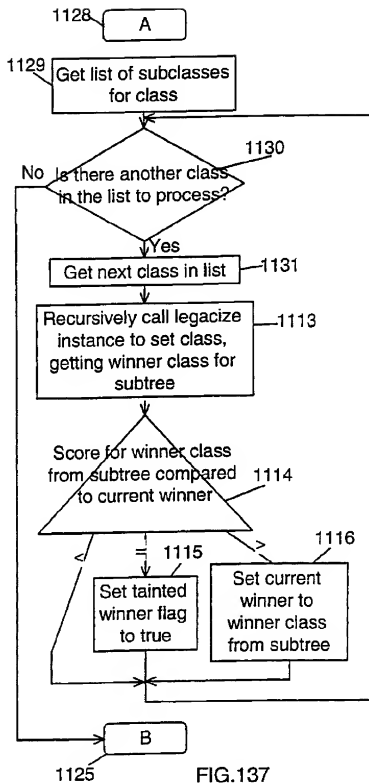


FIG.137

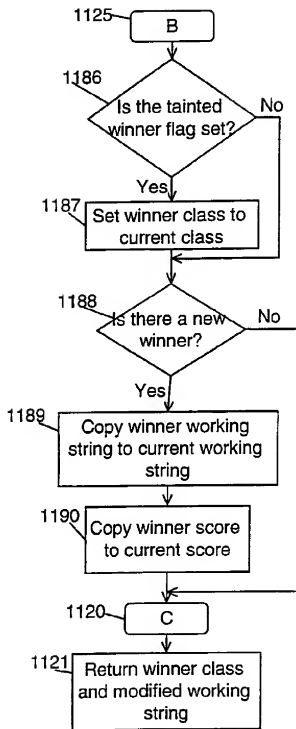


FIG. 138

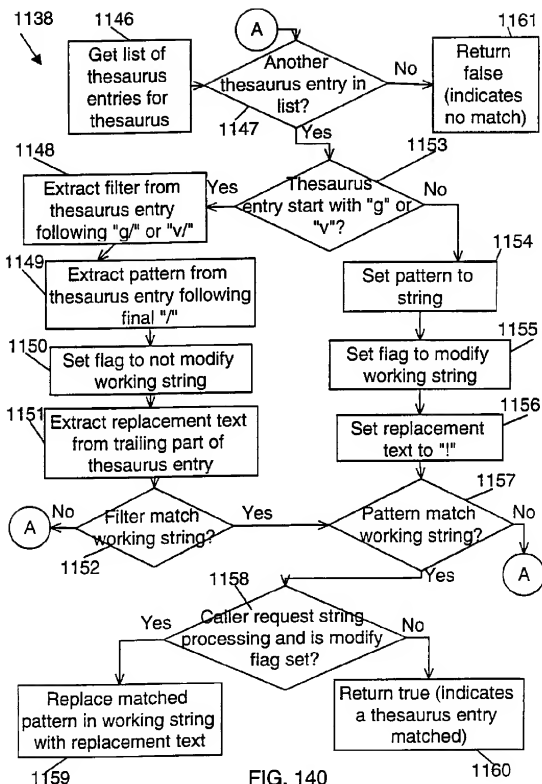


FIG. 140

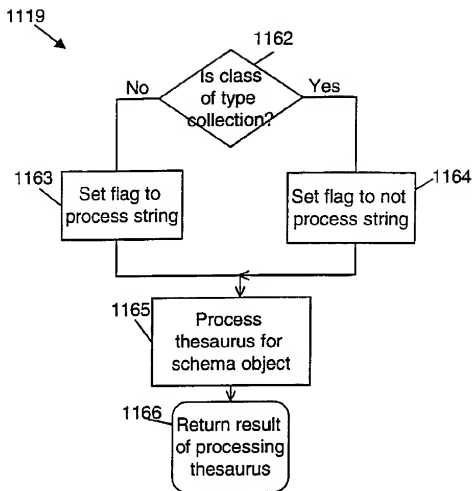


FIG.141

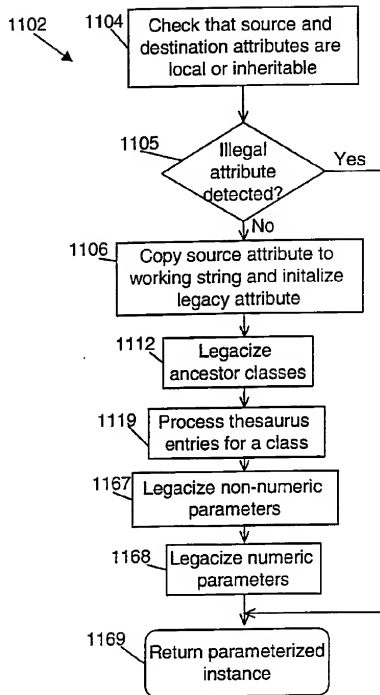


FIG. 142

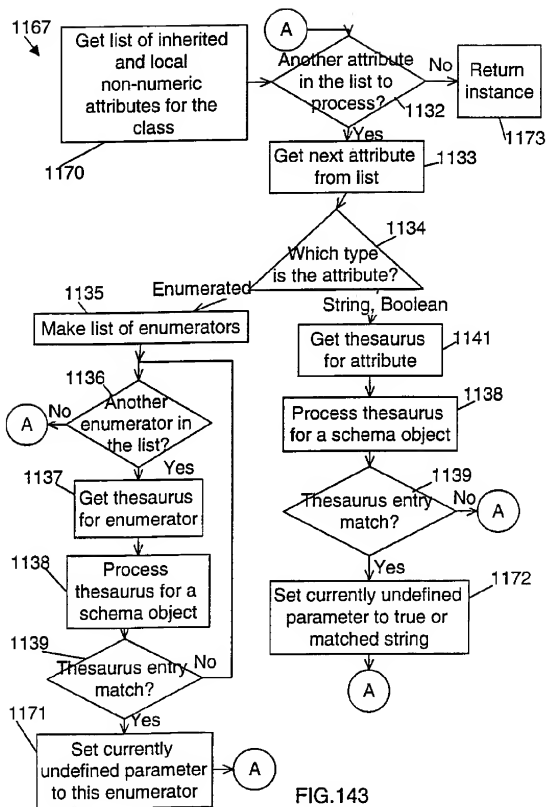


FIG. 143

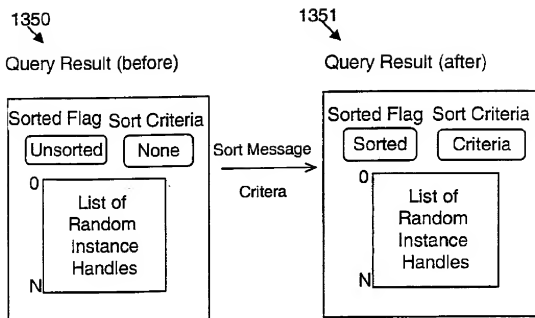


FIG. 144

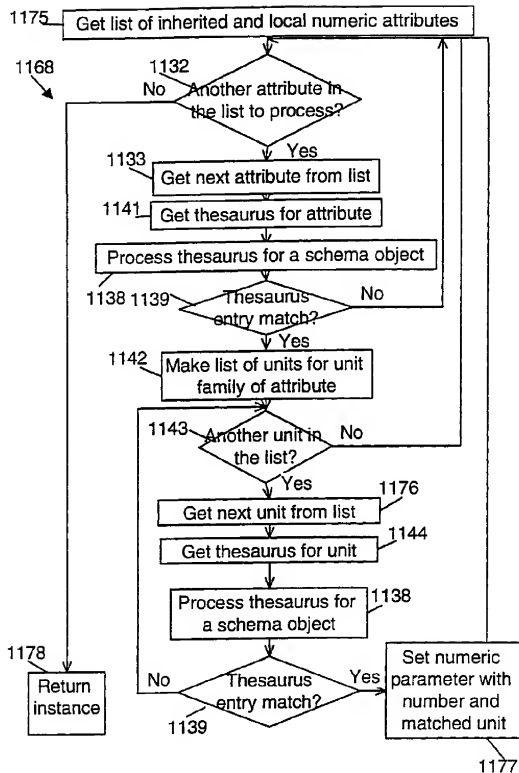


FIG. 145

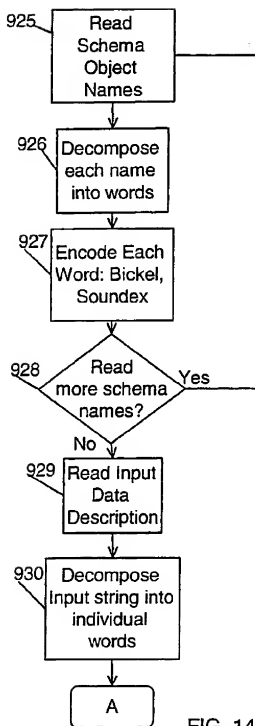


FIG. 146

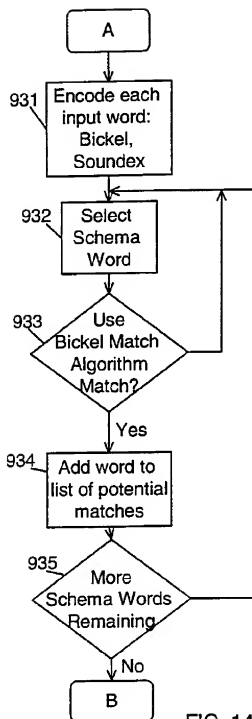


FIG. 147

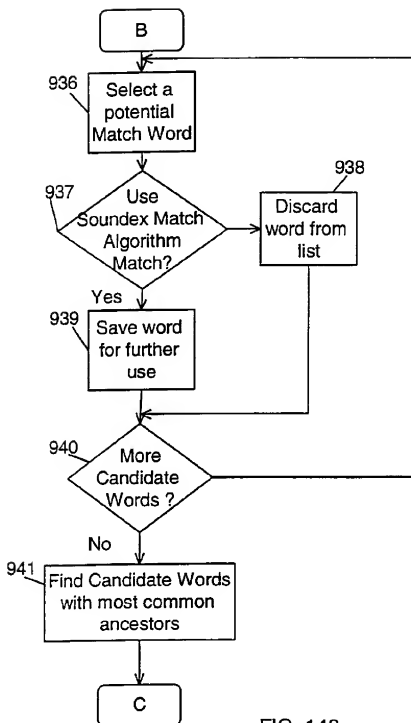


FIG. 148

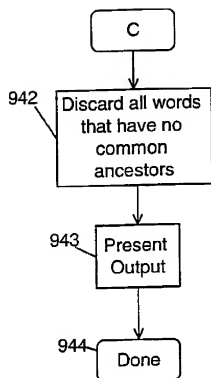


FIG. 149

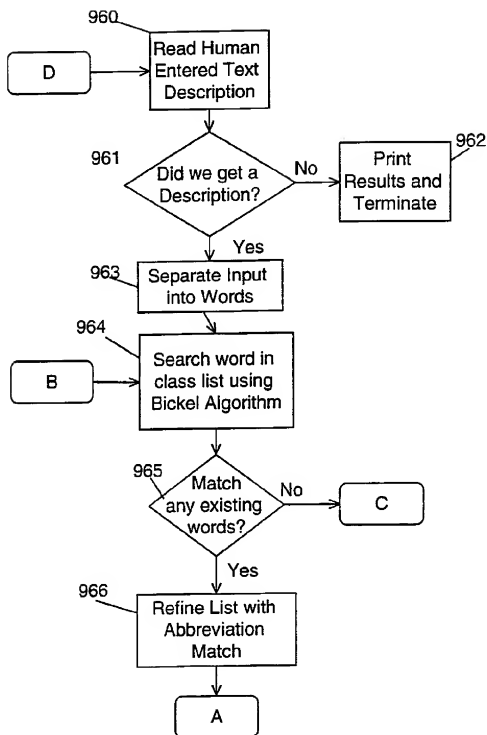


FIG. 150

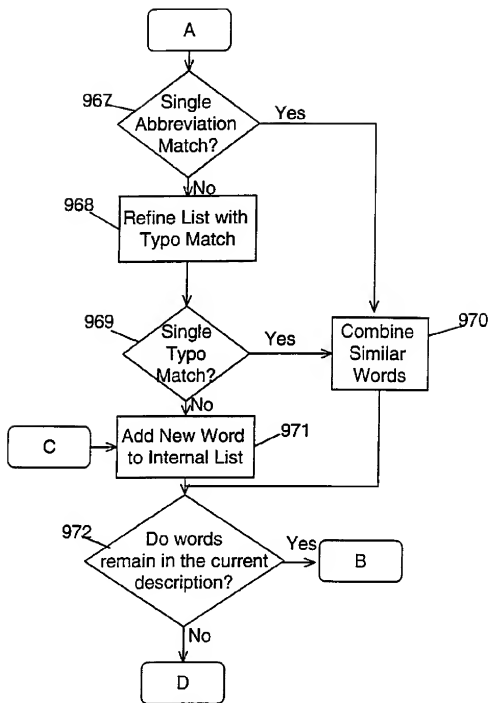


FIG. 151

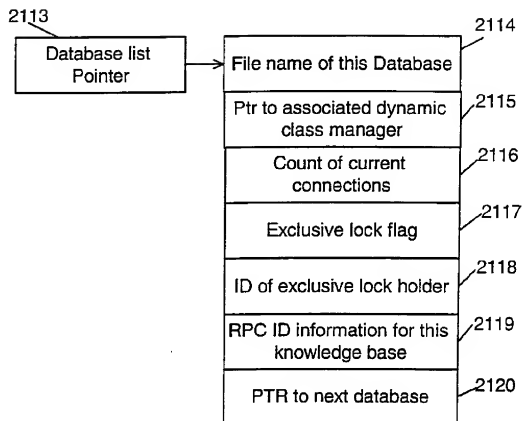


FIG. 152

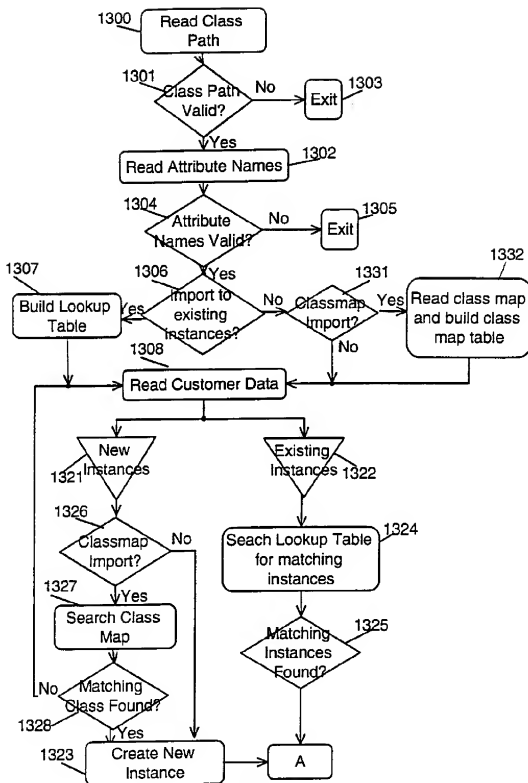


FIG. 153

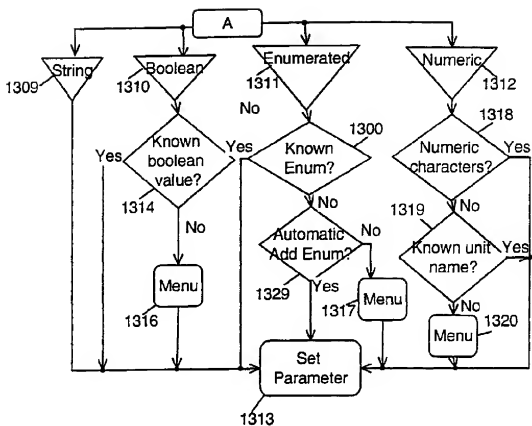


FIG. 154

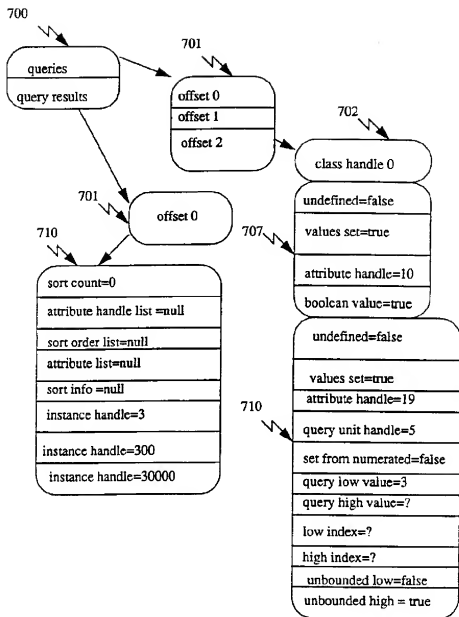


FIG. 155

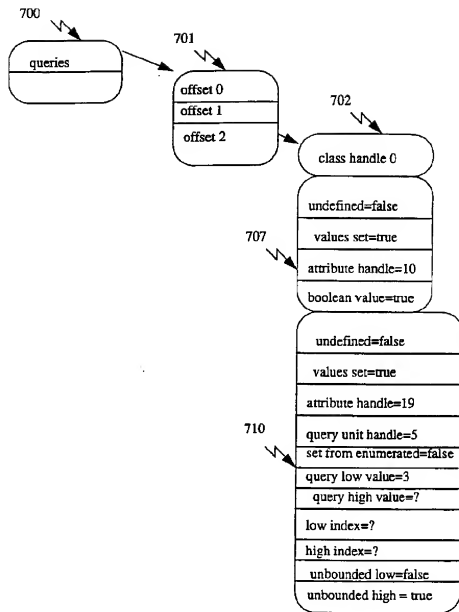


FIG. 156

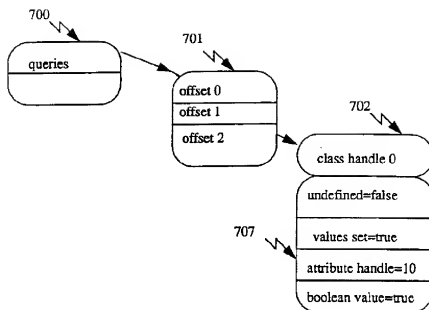


FIG. 157

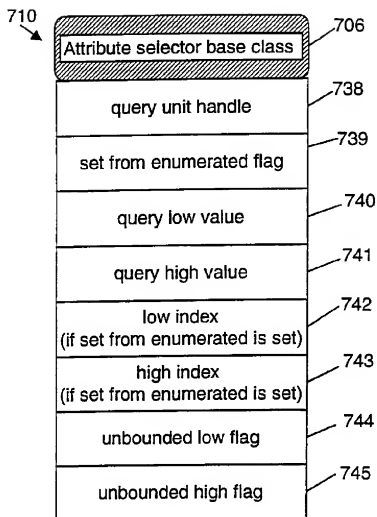


FIG. 158

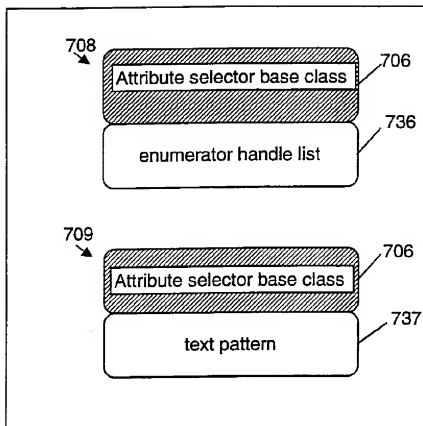


FIG. 159

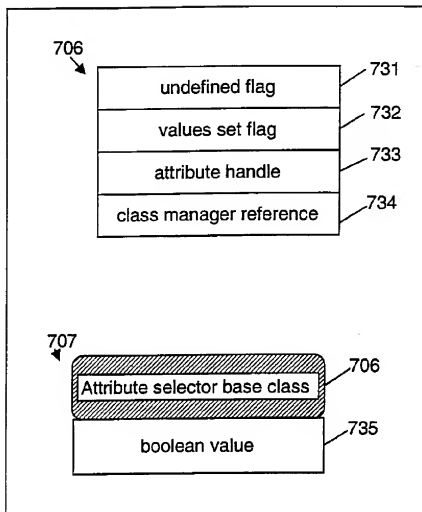


FIG. 160

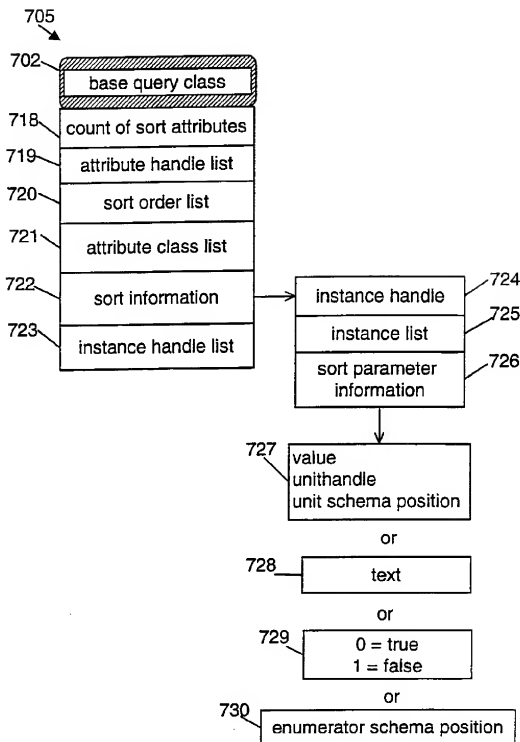


FIG. 161

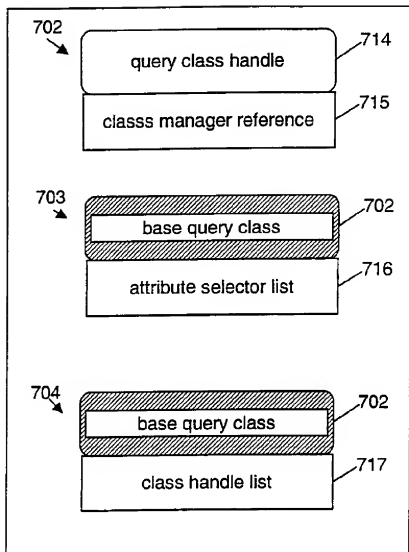


FIG. 162

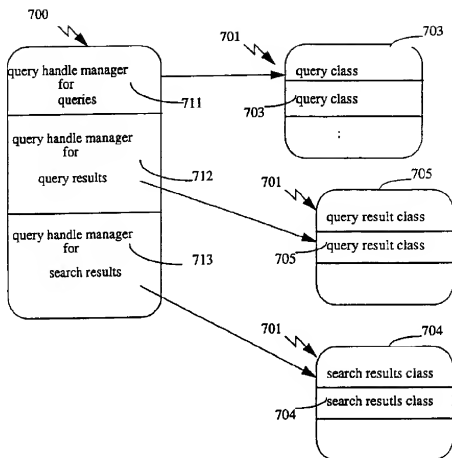


FIG. 163

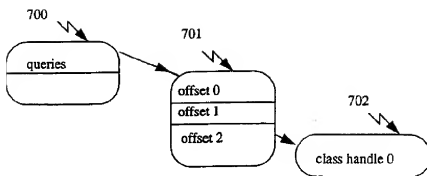


FIG. 164

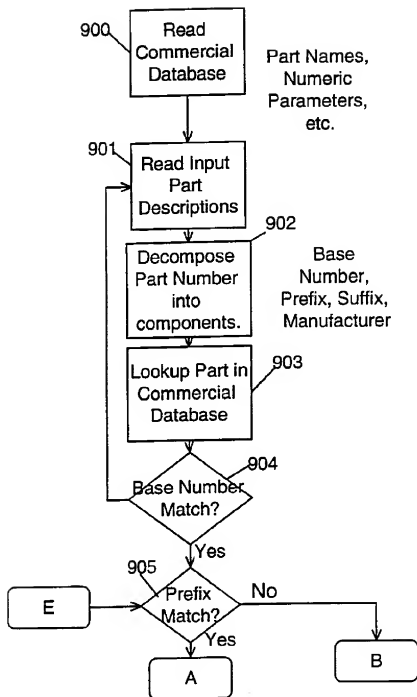


FIG. 165

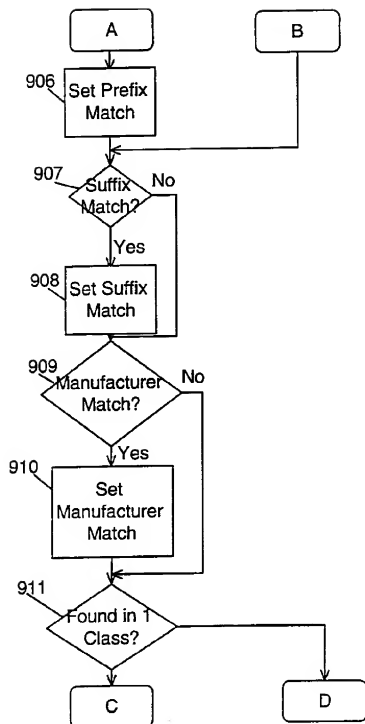


FIG. 166

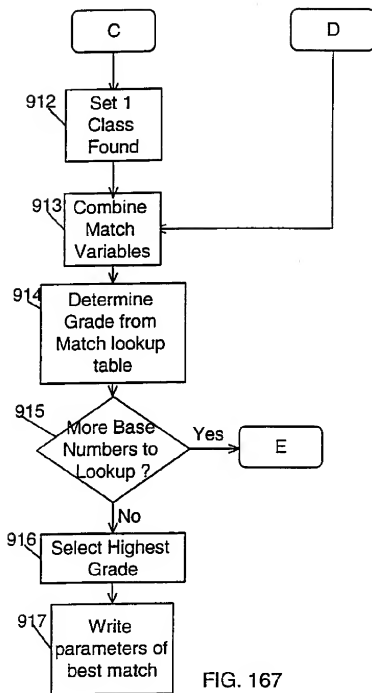


FIG. 167

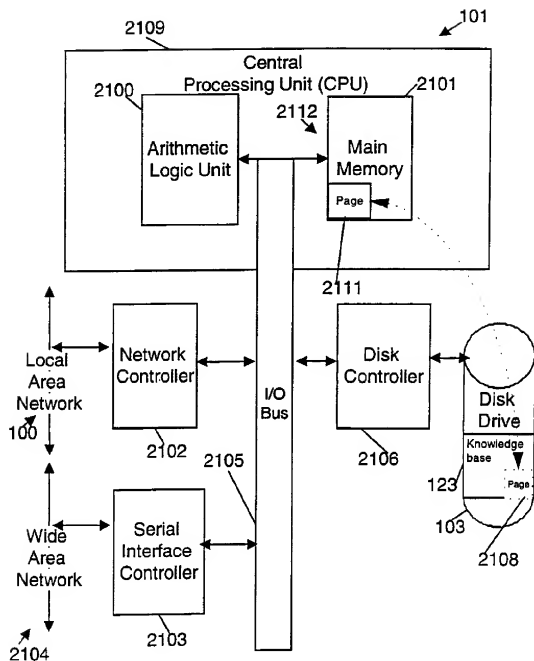


FIG. 168

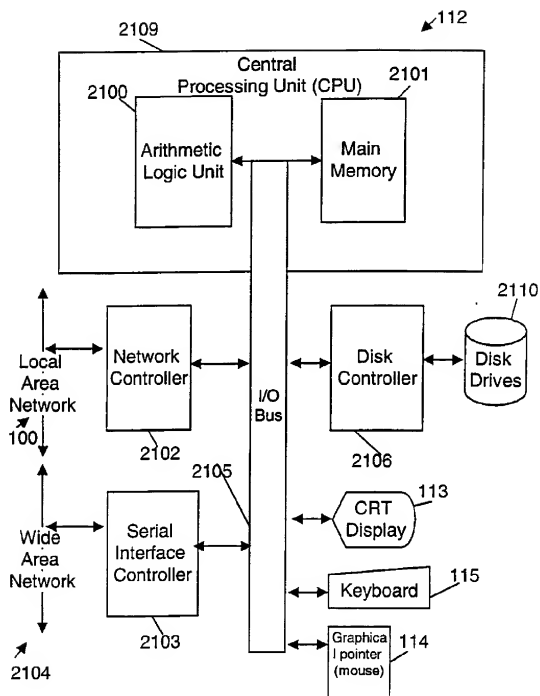


FIG. 169

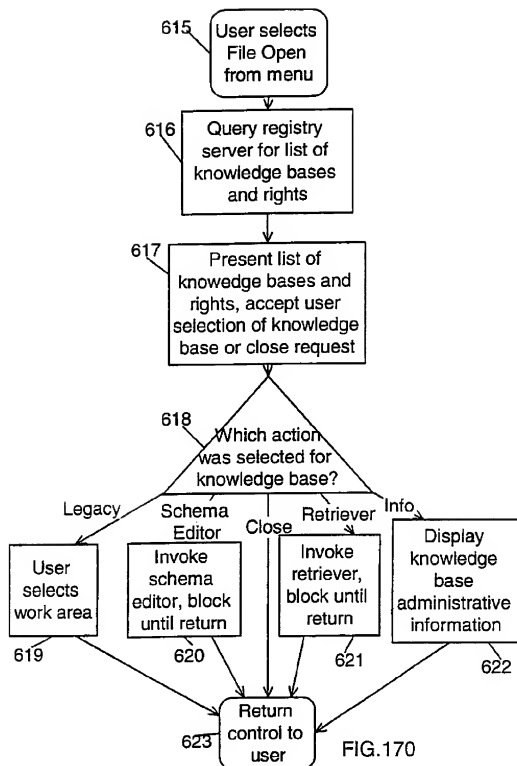




FIG. 171

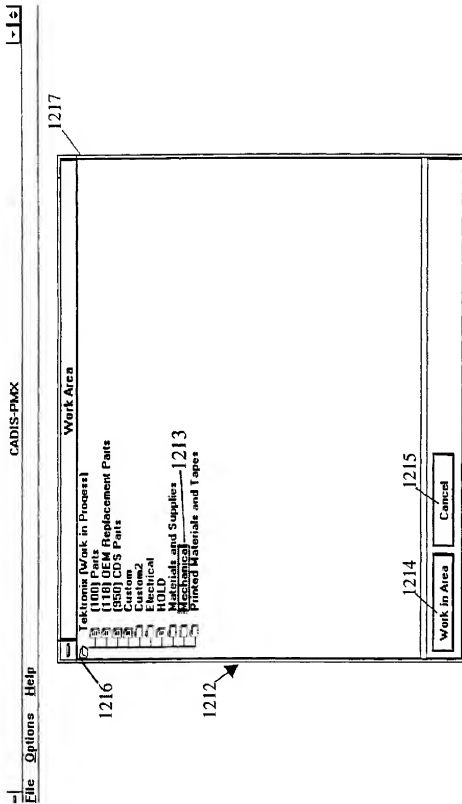


FIG. 172

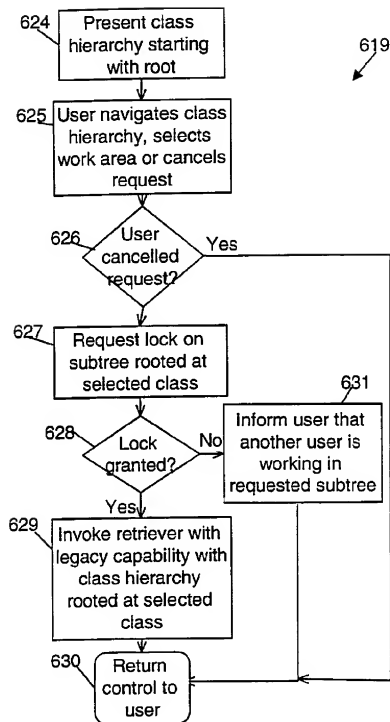


FIG.173

1223 1224 1219

File Legacy Tool CADIS PAK

File Find Options Window Help

1222

Search Criteria

Order	Attribute	Search Criteria
Part Number		T
Part Revision		C
Part Name		T
Description		T
Notes		T
New Design Flag		T
Cost		%
Status		C
Safety Controlled		%
Table Import		%
Technical Classified		%
Preference Flag		%
Spec. on Line		%
MFG Equip Code		T
MFG Part Number		T
MFG Rating		C
MFG Contact Etc.		%

1220

1221

1218

Available Parts: 11779

Set All Clear All Search Criteria Clear All Clear Selected

Display Order

1220

FIG. 174

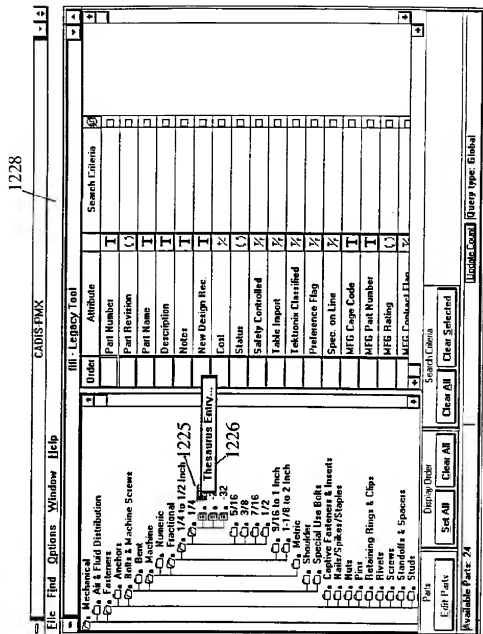


FIG. 175

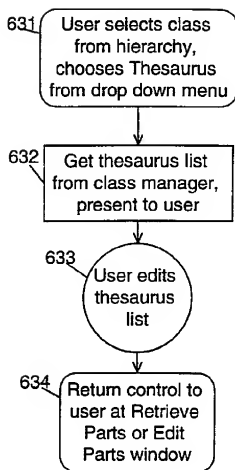


FIG.176

CADIS-PHX III-B Legacy Tool

Elite Find Options Window Help

Search Criteria

Order	Altitude	Search Criteria
1	Part Number	T
2	Part Revision	T
3	Part Name	T
4	Description	T
5	Notes	T
6	New Design Rec.	T
7	Cost	1/2

Thesaurus

Thesaurus Entry

1/14/88 - 420

1229

1230

1233

1234

1235

OK **Cancel** **Add** **Insert** **Delete**

Search Criteria

Display Order

Clear All **Clear Selected**

1227

1231

1229

1230

1233

1234

1235

OK **Cancel** **Add** **Insert** **Delete**

Search Criteria

Display Order

Clear All **Clear Selected**

1227

1231

1229

1230

1233

1234

1235

OK **Cancel** **Add** **Insert** **Delete**

Search Criteria

Display Order

Clear All **Clear Selected**

1227

1231

1229

1230

1233

1234

1235

OK **Cancel** **Add** **Insert** **Delete**

Search Criteria

Display Order

Clear All **Clear Selected**

1227

1231

1229

1230

1233

1234

1235

OK **Cancel** **Add** **Insert** **Delete**

Search Criteria

Display Order

Clear All **Clear Selected**

1227

1231

1229

1230

1233

1234

1235

OK **Cancel** **Add** **Insert** **Delete**

Search Criteria

Display Order

Clear All **Clear Selected**

1227

1231

1229

1230

1233

1234

1235

OK **Cancel** **Add** **Insert** **Delete**

Search Criteria

Display Order

Clear All **Clear Selected**

1227

1231

1229

1230

1233

1234

1235

OK **Cancel** **Add** **Insert** **Delete**

Search Criteria

Display Order

Clear All **Clear Selected**

1227

1231

1229

1230

1233

1234

1235

OK **Cancel** **Add** **Insert** **Delete**

Search Criteria

Display Order

Clear All **Clear Selected**

1227

1231

1229

1230

1233

1234

1235

OK **Cancel** **Add** **Insert** **Delete**

Search Criteria

Display Order

Clear All **Clear Selected**

1227

1231

1229

1230

1233

1234

1235

OK **Cancel** **Add** **Insert** **Delete**

Search Criteria

Display Order

Clear All **Clear Selected**

1227

1231

1229

1230

1233

1234

1235

OK **Cancel** **Add** **Insert** **Delete**

Search Criteria

Display Order

Clear All **Clear Selected**

1227

1231

1229

1230

1233

1234

1235

OK **Cancel** **Add** **Insert** **Delete**

Search Criteria

Display Order

Clear All **Clear Selected**

1227

1231

1229

1230

1233

1234

1235

OK **Cancel** **Add** **Insert** **Delete**

Search Criteria

Display Order

Clear All **Clear Selected**

1227

1231

1229

1230

1233

1234

1235

OK **Cancel** **Add** **Insert** **Delete**

Search Criteria

Display Order

Clear All **Clear Selected**

1227

1231

1229

1230

1233

1234

1235

OK **Cancel** **Add** **Insert** **Delete**

Search Criteria

Display Order

Clear All **Clear Selected**

1227

1231

1229

1230

1233

1234

1235

OK **Cancel** **Add** **Insert** **Delete**

Search Criteria

Display Order

Clear All **Clear Selected**

1227

1231

1229

1230

1233

1234

1235

OK **Cancel** **Add** **Insert** **Delete**

Search Criteria

Display Order

Clear All **Clear Selected**

1227

1231

1229

1230

1233

1234

1235

OK **Cancel** **Add** **Insert** **Delete**

Search Criteria

Display Order

Clear All **Clear Selected**

1227

1231

1229

1230

1233

1234

1235

OK **Cancel** **Add** **Insert** **Delete**

Search Criteria

Display Order

Clear All **Clear Selected**

1227

1231

1229

1230

FIG. 177

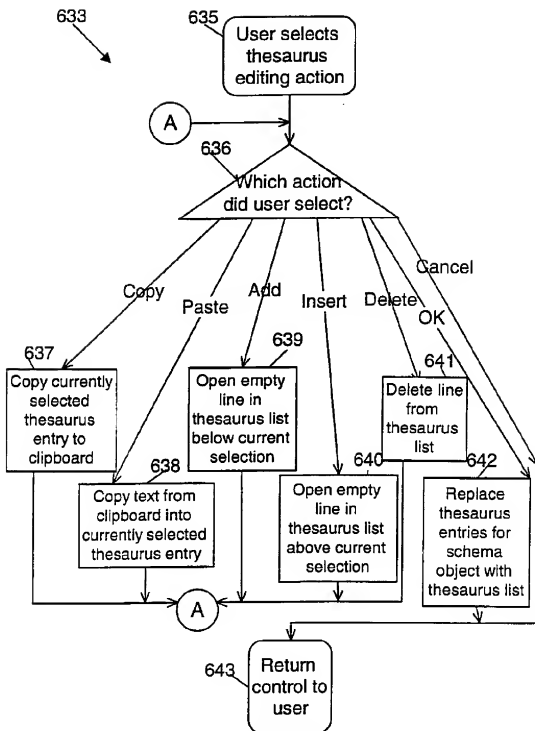


FIG.178

FIG. 179

CAUIS PMX - [III - Legacy Tool]			
Elite Find Options Window Help			
Copy	Part	Order	Search Criteria
Mechanical <input type="checkbox"/> Air & Fluid Distribution <input type="checkbox"/> Fasteners <input type="checkbox"/> Anchors <input type="checkbox"/> Bolts & Machine Screws <input type="checkbox"/> Chains <input type="checkbox"/> Machine <input type="checkbox"/> Numeric <input type="checkbox"/> Fractional <input type="checkbox"/> 1/4 to 1/2 inch <input type="checkbox"/> 3/4 to 1 inch <input type="checkbox"/> 1 1/2 to 2 inch <input type="checkbox"/> 2 to 3 inch <input type="checkbox"/> 3 to 4 inch <input type="checkbox"/> 4 to 5 inch <input type="checkbox"/> 5 to 6 inch <input type="checkbox"/> 6 to 7 inch <input type="checkbox"/> 7 to 8 inch <input type="checkbox"/> 8 to 9 inch <input type="checkbox"/> 9 to 10 inch <input type="checkbox"/> 10 to 11 inch <input type="checkbox"/> 11 to 12 inch <input type="checkbox"/> 12 to 13 inch <input type="checkbox"/> 13 to 14 inch <input type="checkbox"/> 14 to 15 inch <input type="checkbox"/> 15 to 16 inch <input type="checkbox"/> 16 to 17 inch <input type="checkbox"/> 17 to 18 inch <input type="checkbox"/> 18 to 19 inch <input type="checkbox"/> 19 to 20 inch <input type="checkbox"/> 20 to 21 inch <input type="checkbox"/> 21 to 22 inch <input type="checkbox"/> 22 to 23 inch <input type="checkbox"/> 23 to 24 inch <input type="checkbox"/> 24 to 25 inch <input type="checkbox"/> 25 to 26 inch <input type="checkbox"/> 26 to 27 inch <input type="checkbox"/> 27 to 28 inch <input type="checkbox"/> 28 to 29 inch <input type="checkbox"/> 29 to 30 inch <input type="checkbox"/> 30 to 31 inch <input type="checkbox"/> 31 to 32 inch <input type="checkbox"/> 32 to 33 inch <input type="checkbox"/> 33 to 34 inch <input type="checkbox"/> 34 to 35 inch <input type="checkbox"/> 35 to 36 inch <input type="checkbox"/> 36 to 37 inch <input type="checkbox"/> 37 to 38 inch <input type="checkbox"/> 38 to 39 inch <input type="checkbox"/> 39 to 40 inch <input type="checkbox"/> 40 to 41 inch <input type="checkbox"/> 41 to 42 inch <input type="checkbox"/> 42 to 43 inch <input type="checkbox"/> 43 to 44 inch <input type="checkbox"/> 44 to 45 inch <input type="checkbox"/> 45 to 46 inch <input type="checkbox"/> 46 to 47 inch <input type="checkbox"/> 47 to 48 inch <input type="checkbox"/> 48 to 49 inch <input type="checkbox"/> 49 to 50 inch <input type="checkbox"/> 50 to 51 inch <input type="checkbox"/> 51 to 52 inch <input type="checkbox"/> 52 to 53 inch <input type="checkbox"/> 53 to 54 inch <input type="checkbox"/> 54 to 55 inch <input type="checkbox"/> 55 to 56 inch <input type="checkbox"/> 56 to 57 inch <input type="checkbox"/> 57 to 58 inch <input type="checkbox"/> 58 to 59 inch <input type="checkbox"/> 59 to 60 inch <input type="checkbox"/> 60 to 61 inch <input type="checkbox"/> 61 to 62 inch <input type="checkbox"/> 62 to 63 inch <input type="checkbox"/> 63 to 64 inch <input type="checkbox"/> 64 to 65 inch <input type="checkbox"/> 65 to 66 inch <input type="checkbox"/> 66 to 67 inch <input type="checkbox"/> 67 to 68 inch <input type="checkbox"/> 68 to 69 inch <input type="checkbox"/> 69 to 70 inch <input type="checkbox"/> 70 to 71 inch <input type="checkbox"/> 71 to 72 inch <input type="checkbox"/> 72 to 73 inch <input type="checkbox"/> 73 to 74 inch <input type="checkbox"/> 74 to 75 inch <input type="checkbox"/> 75 to 76 inch <input type="checkbox"/> 76 to 77 inch <input type="checkbox"/> 77 to 78 inch <input type="checkbox"/> 78 to 79 inch <input type="checkbox"/> 79 to 80 inch <input type="checkbox"/> 80 to 81 inch <input type="checkbox"/> 81 to 82 inch <input type="checkbox"/> 82 to 83 inch <input type="checkbox"/> 83 to 84 inch <input type="checkbox"/> 84 to 85 inch <input type="checkbox"/> 85 to 86 inch <input type="checkbox"/> 86 to 87 inch <input type="checkbox"/> 87 to 88 inch <input type="checkbox"/> 88 to 89 inch <input type="checkbox"/> 89 to 90 inch <input type="checkbox"/> 90 to 91 inch <input type="checkbox"/> 91 to 92 inch <input type="checkbox"/> 92 to 93 inch <input type="checkbox"/> 93 to 94 inch <input type="checkbox"/> 94 to 95 inch <input type="checkbox"/> 95 to 96 inch <input type="checkbox"/> 96 to 97 inch <input type="checkbox"/> 97 to 98 inch <input type="checkbox"/> 98 to 99 inch <input type="checkbox"/> 99 to 100 inch <input type="checkbox"/> 100 to 101 inch <input type="checkbox"/> 101 to 102 inch <input type="checkbox"/> 102 to 103 inch <input type="checkbox"/> 103 to 104 inch <input type="checkbox"/> 104 to 105 inch <input type="checkbox"/> 105 to 106 inch <input type="checkbox"/> 106 to 107 inch <input type="checkbox"/> 107 to 108 inch <input type="checkbox"/> 108 to 109 inch <input type="checkbox"/> 109 to 110 inch <input type="checkbox"/> 110 to 111 inch <input type="checkbox"/> 111 to 112 inch <input type="checkbox"/> 112 to 113 inch <input type="checkbox"/> 113 to 114 inch <input type="checkbox"/> 114 to 115 inch <input type="checkbox"/> 115 to 116 inch <input type="checkbox"/> 116 to 117 inch <input type="checkbox"/> 117 to 118 inch <input type="checkbox"/> 118 to 119 inch <input type="checkbox"/> 119 to 120 inch <input type="checkbox"/> 120 to 121 inch <input type="checkbox"/> 121 to 122 inch <input type="checkbox"/> 122 to 123 inch <input type="checkbox"/> 123 to 124 inch <input type="checkbox"/> 124 to 125 inch <input type="checkbox"/> 125 to 126 inch <input type="checkbox"/> 126 to 127 inch <input type="checkbox"/> 127 to 128 inch <input type="checkbox"/> 128 to 129 inch <input type="checkbox"/> 129 to 130 inch <input type="checkbox"/> 130 to 131 inch <input type="checkbox"/> 131 to 132 inch <input type="checkbox"/> 132 to 133 inch <input type="checkbox"/> 133 to 134 inch <input type="checkbox"/> 134 to 135 inch <input type="checkbox"/> 135 to 136 inch <input type="checkbox"/> 136 to 137 inch <input type="checkbox"/> 137 to 138 inch <input type="checkbox"/> 138 to 139 inch <input type="checkbox"/> 139 to 140 inch <input type="checkbox"/> 140 to 141 inch <input type="checkbox"/> 141 to 142 inch <input type="checkbox"/> 142 to 143 inch <input type="checkbox"/> 143 to 144 inch <input type="checkbox"/> 144 to 145 inch <input type="checkbox"/> 145 to 146 inch <input type="checkbox"/> 146 to 147 inch <input type="checkbox"/> 147 to 148 inch <input type="checkbox"/> 148 to 149 inch <input type="checkbox"/> 149 to 150 inch <input type="checkbox"/> 150 to 151 inch <input type="checkbox"/> 151 to 152 inch <input type="checkbox"/> 152 to 153 inch <input type="checkbox"/> 153 to 154 inch <input type="checkbox"/> 154 to 155 inch <input type="checkbox"/> 155 to 156 inch <input type="checkbox"/> 156 to 157 inch <input type="checkbox"/> 157 to 158 inch <input type="checkbox"/> 158 to 159 inch <input type="checkbox"/> 159 to 160 inch <input type="checkbox"/> 160 to 161 inch <input type="checkbox"/> 161 to 162 inch <input type="checkbox"/> 162 to 163 inch <input type="checkbox"/> 163 to 164 inch <input type="checkbox"/> 164 to 165 inch <input type="checkbox"/> 165 to 166 inch <input type="checkbox"/> 166 to 167 inch <input type="checkbox"/> 167 to 168 inch <input type="checkbox"/> 168 to 169 inch <input type="checkbox"/> 169 to 170 inch <input type="checkbox"/> 170 to 171 inch <input type="checkbox"/> 171 to 172 inch <input type="checkbox"/> 172 to 173 inch <input type="checkbox"/> 173 to 174 inch <input type="checkbox"/> 174 to 175 inch <input type="checkbox"/> 175 to 176 inch <input type="checkbox"/> 176 to 177 inch <input type="checkbox"/> 177 to 178 inch <input type="checkbox"/> 178 to 179 inch <input type="checkbox"/> 179 to 180 inch <input type="checkbox"/> 180 to 181 inch <input type="checkbox"/> 181 to 182 inch <input type="checkbox"/> 182 to 183 inch <input type="checkbox"/> 183 to 184 inch <input type="checkbox"/> 184 to 185 inch <input type="checkbox"/> 185 to 186 inch <input type="checkbox"/> 186 to 187 inch <input type="checkbox"/> 187 to 188 inch <input type="checkbox"/> 188 to 189 inch <input type="checkbox"/> 189 to 190 inch <input type="checkbox"/> 190 to 191 inch <input type="checkbox"/> 191 to 192 inch <input type="checkbox"/> 192 to 193 inch <input type="checkbox"/> 193 to 194 inch <input type="checkbox"/> 194 to 195 inch <input type="checkbox"/> 195 to 196 inch <input type="checkbox"/> 196 to 197 inch <input type="checkbox"/> 197 to 198 inch <input type="checkbox"/> 198 to 199 inch <input type="checkbox"/> 199 to 200 inch <input type="checkbox"/> 200 to 201 inch <input type="checkbox"/> 201 to 202 inch <input type="checkbox"/> 202 to 203 inch <input type="checkbox"/> 203 to 204 inch <input type="checkbox"/> 204 to 205 inch <input type="checkbox"/> 205 to 206 inch <input type="checkbox"/> 206 to 207 inch <input type="checkbox"/> 207 to 208 inch <input type="checkbox"/> 208 to 209 inch <input type="checkbox"/> 209 to 210 inch <input type="checkbox"/> 210 to 211 inch <input type="checkbox"/> 211 to 212 inch <input type="checkbox"/> 212 to 213 inch <input type="checkbox"/> 213 to 214 inch <input type="checkbox"/> 214 to 215 inch <input type="checkbox"/> 215 to 216 inch <input type="checkbox"/> 216 to 217 inch <input type="checkbox"/> 217 to 218 inch <input type="checkbox"/> 218 to 219 inch <input type="checkbox"/> 219 to 220 inch <input type="checkbox"/> 220 to 221 inch <input type="checkbox"/> 221 to 222 inch <input type="checkbox"/> 222 to 223 inch <input type="checkbox"/> 223 to 224 inch <input type="checkbox"/> 224 to 225 inch <input type="checkbox"/> 225 to 22			

FIG. 180

FIG. 181

CADIS PMX - [iii] - Legacy Tool

File Find Options Window Help

Mechanical

☒ As & Fluid Distribution
☒ Assemblies
☒ Anchors
☒ Bells & Machine Screws
☒ Brackets
☒ Bushings
☒ Caps
☒ Chokes
☒ Clamps
☒ Connectors
☒ Couplings
☒ Drums
☒ Flanges
☒ Fittings
☒ Gaskets
☒ Hangers
☒ Hoses
☒ Joints
☒ Manifolds
☒ Nozzles
☒ Orifices
☒ Pipes
☒ Plugs
☒ Pumps
☒ Seals
☒ Shims
☒ Spacers
☒ Studs
☒ Taps
☒ Valves
☒ Washers
☒ Welds
☒ Wire
☒ Miscellaneous
☒ Modular Packaging System
☒ Optics

☒ 1/4 to 1/2 Inch
☒ 1/2 to 1 Inch
☒ 1 to 2 Inch
☒ 2 to 4 Inch
☒ 4 to 8 Inch
☒ 8 to 16 Inch
☒ 16 to 32 Inch
☒ 32 to 64 Inch
☒ 64 to 128 Inch
☒ 128 to 256 Inch
☒ 256 to 512 Inch
☒ 512 to 1024 Inch
☒ 1024 to 2048 Inch
☒ 2048 to 4096 Inch
☒ 4096 to 8192 Inch
☒ 8192 to 16384 Inch
☒ 16384 to 32768 Inch
☒ 32768 to 65536 Inch
☒ 65536 to 131072 Inch
☒ 131072 to 262144 Inch
☒ 262144 to 524288 Inch
☒ 524288 to 1048576 Inch
☒ 1048576 to 2097152 Inch
☒ 2097152 to 4194304 Inch
☒ 4194304 to 8388608 Inch
☒ 8388608 to 16777216 Inch
☒ 16777216 to 33554432 Inch
☒ 33554432 to 67108864 Inch
☒ 67108864 to 134217728 Inch
☒ 134217728 to 268435456 Inch
☒ 268435456 to 536870912 Inch
☒ 536870912 to 1073741824 Inch
☒ 1073741824 to 2147483648 Inch
☒ 2147483648 to 4294967296 Inch
☒ 4294967296 to 8589934592 Inch
☒ 8589934592 to 17179869184 Inch
☒ 17179869184 to 34359738368 Inch
☒ 34359738368 to 68719476736 Inch
☒ 68719476736 to 137438953472 Inch
☒ 137438953472 to 274877906944 Inch
☒ 274877906944 to 549755813888 Inch
☒ 549755813888 to 1099511627776 Inch
☒ 1099511627776 to 2199023255552 Inch
☒ 2199023255552 to 4398046511104 Inch
☒ 4398046511104 to 8796093022208 Inch
☒ 8796093022208 to 17592186044416 Inch
☒ 17592186044416 to 35184372088832 Inch
☒ 35184372088832 to 70368744177664 Inch
☒ 70368744177664 to 140737488355328 Inch
☒ 140737488355328 to 281474976710656 Inch
☒ 281474976710656 to 562949953421312 Inch
☒ 562949953421312 to 1125899906842624 Inch
☒ 1125899906842624 to 2251799813685248 Inch
☒ 2251799813685248 to 4503599627370496 Inch
☒ 4503599627370496 to 9007199254740992 Inch
☒ 9007199254740992 to 18014398509481984 Inch
☒ 18014398509481984 to 36028797018963968 Inch
☒ 36028797018963968 to 72057594037927936 Inch
☒ 72057594037927936 to 144115188075855872 Inch
☒ 144115188075855872 to 288230376151711744 Inch
☒ 288230376151711744 to 576460752303423488 Inch
☒ 576460752303423488 to 1152921504606846976 Inch
☒ 1152921504606846976 to 2305843009213693952 Inch
☒ 2305843009213693952 to 4611686018427387904 Inch
☒ 4611686018427387904 to 9223372036854775808 Inch
☒ 9223372036854775808 to 18446744073709551616 Inch
☒ 18446744073709551616 to 36893488147419103232 Inch
☒ 36893488147419103232 to 73786976294838206464 Inch
☒ 73786976294838206464 to 147573952589676412928 Inch
☒ 147573952589676412928 to 295147905179352825856 Inch
☒ 295147905179352825856 to 590295810358705651712 Inch
☒ 590295810358705651712 to 1180591620717411303424 Inch
☒ 1180591620717411303424 to 2361183241434822606848 Inch
☒ 2361183241434822606848 to 4722366482869645213696 Inch
☒ 4722366482869645213696 to 9444732965739290427392 Inch
☒ 9444732965739290427392 to 18889465931478580854784 Inch
☒ 18889465931478580854784 to 37778931862957161709568 Inch
☒ 37778931862957161709568 to 75557863725914323419136 Inch
☒ 75557863725914323419136 to 151115727451828646838272 Inch
☒ 151115727451828646838272 to 302231454903657293676544 Inch
☒ 302231454903657293676544 to 604462909807314587353088 Inch
☒ 604462909807314587353088 to 1208925819614629174706176 Inch
☒ 1208925819614629174706176 to 2417851639229258349412352 Inch
☒ 2417851639229258349412352 to 4835703278458516698824704 Inch
☒ 4835703278458516698824704 to 9671406556917033397649408 Inch
☒ 9671406556917033397649408 to 19342813113834066795298816 Inch
☒ 19342813113834066795298816 to 38685626227668133590597632 Inch
☒ 38685626227668133590597632 to 77371252455336267181195264 Inch
☒ 77371252455336267181195264 to 154742504910672534362390528 Inch
☒ 154742504910672534362390528 to 309485009821345068724781056 Inch
☒ 309485009821345068724781056 to 618970019642690137449562112 Inch
☒ 618970019642690137449562112 to 1237940039285380274899124224 Inch
☒ 1237940039285380274899124224 to 2475880078570760549798248448 Inch
☒ 2475880078570760549798248448 to 4951760157141521099596496896 Inch
☒ 4951760157141521099596496896 to 9903520314283042199192993792 Inch
☒ 9903520314283042199192993792 to 19807040628566084398385987584 Inch
☒ 19807040628566084398385987584 to 39614081257132168796771975168 Inch
☒ 39614081257132168796771975168 to 79228162514264337593543950336 Inch
☒ 79228162514264337593543950336 to 158456325028528675187087900672 Inch
☒ 158456325028528675187087900672 to 316912650057057350374175801344 Inch
☒ 316912650057057350374175801344 to 633825300114114700748351602688 Inch
☒ 633825300114114700748351602688 to 1267650600228229401496703205376 Inch
☒ 1267650600228229401496703205376 to 2535301200456458802993406410752 Inch
☒ 2535301200456458802993406410752 to 5070602400912917605986812821504 Inch
☒ 5070602400912917605986812821504 to 10141204801825835211973625643008 Inch
☒ 10141204801825835211973625643008 to 20282409603651670423947251286016 Inch
☒ 20282409603651670423947251286016 to 40564819207303340847894502572032 Inch
☒ 40564819207303340847894502572032 to 81129638414606681695789005144064 Inch
☒ 81129638414606681695789005144064 to 162259276829213363391578010288128 Inch
☒ 162259276829213363391578010288128 to 324518553658426726783156020576256 Inch
☒ 324518553658426726783156020576256 to 649037107316853453566312041152512 Inch
☒ 649037107316853453566312041152512 to 1298074214633706907132624082305024 Inch
☒ 1298074214633706907132624082305024 to 2596148429267413814265248164610048 Inch
☒ 2596148429267413814265248164610048 to 5192296858534827628530496329220096 Inch
☒ 5192296858534827628530496329220096 to 10384593717069655257060992658440192 Inch
☒ 10384593717069655257060992658440192 to 20769187434139310514121985316880384 Inch
☒ 20769187434139310514121985316880384 to 41538374868278621028243970633760768 Inch
☒ 41538374868278621028243970633760768 to 83076749736557242056487941267521536 Inch
☒ 83076749736557242056487941267521536 to 166153499473114484112975882535043072 Inch
☒ 166153499473114484112975882535043072 to 332306998946228968225951765070086144 Inch
☒ 332306998946228968225951765070086144 to 664613997892457936451903530140172288 Inch
☒ 664613997892457936451903530140172288 to 1329227995784915872903807060280344576 Inch
☒ 1329227995784915872903807060280344576 to 2658455991569831745807614120560689152 Inch
☒ 2658455991569831745807614120560689152 to 5316911983139663491615228241121378304 Inch
☒ 5316911983139663491615228241121378304 to 10633823966279326983230456482242756608 Inch
☒ 10633823966279326983230456482242756608 to 21267647932558653966460912964485513216 Inch
☒ 21267647932558653966460912964485513216 to 42535295865117307932921825928971026432 Inch
☒ 42535295865117307932921825928971026432 to 85070591730234615865843651857942052864 Inch
☒ 85070591730234615865843651857942052864 to 170141183460469231731687303715884105728 Inch
☒ 170141183460469231731687303715884105728 to 340282366920938463463374607431768211456 Inch
☒ 340282366920938463463374607431768211456 to 680564733841876926926749214863536422912 Inch
☒ 680564733841876926926749214863536422912 to 1361129467683753853853498429727072845824 Inch
☒ 1361129467683753853853498429727072845824 to 2722258935367507707706996859454145691648 Inch
☒ 2722258935367507707706996859454145691648 to 5444517870735015415413993718908291383296 Inch
☒ 5444517870735015415413993718908291383296 to 10889035741470030830827987437816582766592 Inch
☒ 10889035741470030830827987437816582766592 to 21778071482940061661655974875633165533184 Inch
☒ 21778071482940061661655974875633165533184 to 43556142965880123323311949751266331066368 Inch
☒ 43556142965880123323311949751266331066368 to 87112285931760246646623899502532662132736 Inch
☒ 87112285931760246646623899502532662132736 to 174224571863520493293247799005065324265472 Inch
☒ 174224571863520493293247799005065324265472 to 348449143727040986586495598010130648530944 Inch
☒ 348449143727040986586495598010130648530944 to 696898287454081973172991196020261297061888 Inch
☒ 696898287454081973172991196020261297061888 to 1393796574908163946345982392040522594123776 Inch
☒ 1393796574908163946345982392040522594123776 to 2787593149816327892691964784081045188247552 Inch
☒ 2787593149816327892691964784081045188247552 to 5575186299632655785383929568162090376495104 Inch
☒ 5575186299632655785383929568162090376495104 to 11150372599265311570767859136324180752990208 Inch
☒ 11150372599265311570767859136324180752990208 to 22300745198530623141535718272648361505980416 Inch
☒ 22300745198530623141535718272648361505980416 to 44601490397061246283071436545296723011960832 Inch
☒ 44601490397061246283071436545296723011960832 to 89202980794122492566142873090593446023921664 Inch
☒ 89202980794122492566142873090593446023921664 to 178405961588244985132285746181186892047843328 Inch
☒ 178405961588244985132285746181186892047843328 to 356811923176489970264571492362373784095686656 Inch
☒ 356811923176489970264571492362373784095686656 to 713623846352979940529142984724747568191373312 Inch
☒ 713623846352979940529142984724747568191373312 to 1427247692705959881058285969449495136382746624 Inch
☒ 1427247692705959881058285969449495136382746624 to 2854495385411919762116571938898990272765493248 Inch
☒ 2854495385411919762116571938898990272765493248 to 5708990770823839524233143877797980545530986496 Inch
☒ 5708990770823839524233143877797980545530986496 to 11417981541647679048466287755595961091061972992 Inch
☒ 11417981541647679048466287755595961091061972992 to 22835963083295358096932575511191922182123945984 Inch
☒ 22835963083295358096932575511191922182123945984 to 45671926166590716193865151022383844364247891968 Inch
☒ 45671926166590716193865151022383844364247891968 to 91343852333181432387730302044767688728495783936 Inch
☒ 91343852333181432387730302044767688728495783936 to 182687704666362864775460604089535377456991567872 Inch
☒ 182687704666362864775460604089535377456991567872 to 365375409332725729550921208179070754913983135744 Inch
☒ 365375409332725729550921208179070754913983135744 to 730750818665451459101842416358141509827966271488 Inch
☒ 730750818665451459101842416358141509827966271488 to 1461501637330902918203684832716283019655932542976 Inch
☒ 1461501637330902918203684832716283019655932542976 to 2923003274661805836407369665432566039311865085952 Inch
☒ 2923003274661805836407369665432566039311865085952 to 5846006549323611672814739330865132078623730171904 Inch
☒ 5846006549323611672814739330865132078623730171904 to 11692013098647223345629478661730264157247460343808 Inch
☒ 11692013098647223345629478661730264157247460343808 to 23384026197294446691258957323460528314494920687616 Inch
☒ 23384026197294446691258957323460528314494920687616 to 46768052394588893382517914646921056628989841375232 Inch
☒ 46768052394588893382517914646921056628989841375232 to 93536104789177786765035829293842113257979682750464 Inch
☒ 93536104789177786765035829293842113257979682750464 to 187072209578355573530071658587684226515959365500928 Inch
☒ 187072209578355573530071658587684226515959365500928 to 374144419156711147060143317175368453031918731001856 Inch
☒ 374144419156711147060143317175368453031918731001856 to 748288838313422294120286634350736906063837462003712 Inch
☒ 748288838313422294120286634350736906063837462003712 to 1496577676626844588240573268701473812127674924007424 Inch
☒ 1496577676626844588240573268701473812127674924007424 to 2993155353253689176481146537402947624255349848014848 Inch
☒ 2993155353253689176481146537402947624255349848014848 to 5986310706507378352962293074805895248510699696029696 Inch
☒ 5986310706507378352962293074805895248510699696029696 to 11972621413014756705924586149611790497021399392059392 Inch
☒ 11972621413014756705924586149611790497021399392059392 to 23945242826029513411849172299223580994042798784118784 Inch
☒ 23945242826029513411849172299223580994042798784118784 to 4

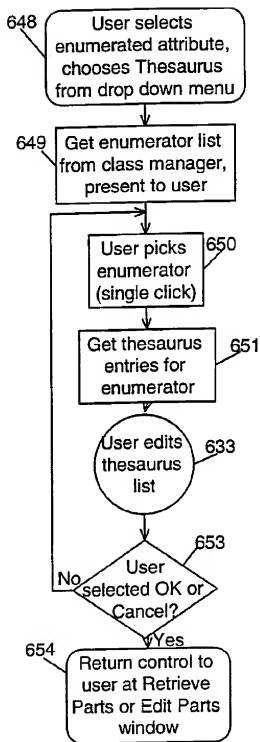


FIG. 184

CADIS PMX - [Full - Legacy Tool]

File Find Options Window Help

Mechanical
☐ Air & Fluid Distribution
☐ Fasteners
☐ Anchors
☐ Bolts & Machine Screws
☐ Brackets
☐ Castings
☐ Machine
☐ Nuts
☐ Numeric
☐ Fractional
☐ 1/4 to 1/2 Inch
☐ 1/2 to 3/4 Inch
☐ 3/4 to 1 Inch
☐ 1 to 1 1/2 Inch
☐ 1 1/2 to 2 Inch
☐ Metric
☐ Rivets
☐ Sealant
☐ Special Use Bolts
☐ Capable Fasteners & Inserts
☐ Nail/Spikes/Staples
☐ Pins
☐ Retaining Rings & Clips
☐ Screws
☐ Standoffs & Spacers
☐ Studs
☐ Washers
☐ General Hardware
☐ Modular Packaging System
☐ Other

Order

Attitude

Tektronics Classified

Preference Flag

Spec. on Line

MFG Caps Exide

MFG Part Number

MFG Rating

MFG Contact Flag

Predicted Supplier

Relocated

Finish

Main Material

Attached Washer

Drilled

Head Recess

Head Style

Left Hand Thread

Self Locking

Shank Type

Length

Search Criteria

1246

1244

Thermax Entry...

Set Order

Set Criteria

Clear Criteria

Search Criteria

Display Order

Set All

Clear All

Clear Selected

Available Parts: 28

Update Count

FIG. 185

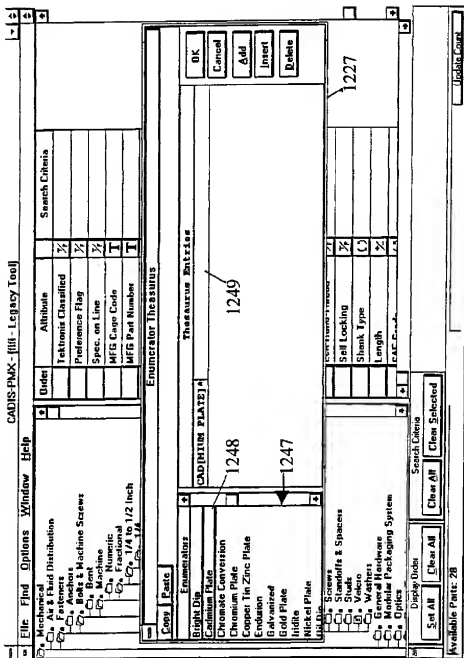


FIG. 186

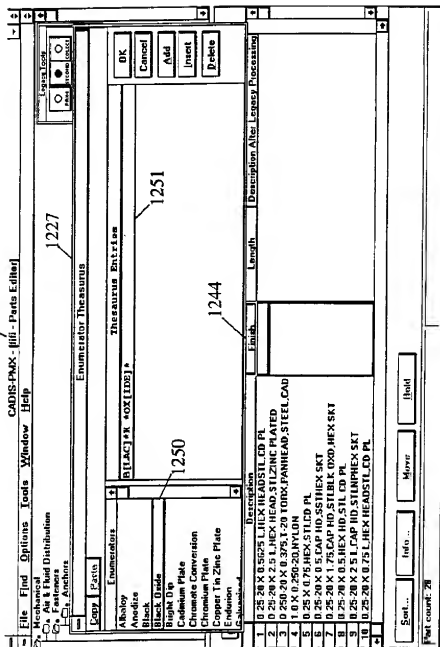


FIG. 187

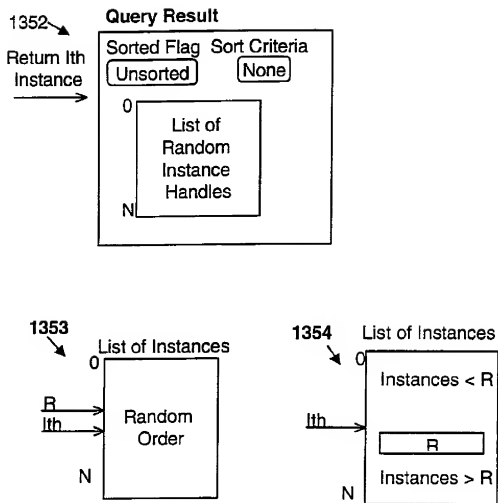


FIG. 188

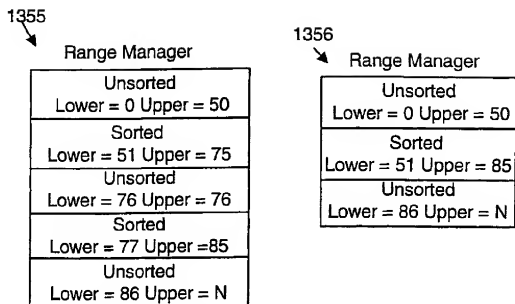


FIG. 189

1243

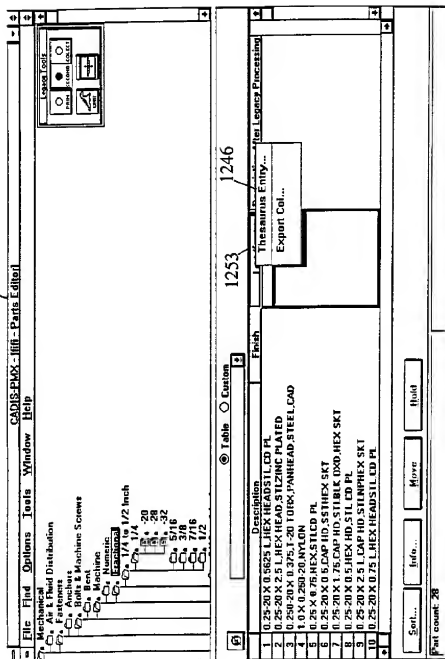


FIG. 190

CADIS-PHX - (Pili - Parts Editor) 1227

File Find Options Tools Window Help

☒ Mechanical
☐ Air & Fluid Distribution
☐ Pumps
☐ Air Duct

Insert Code
☐ Add
☐ Remove

Thesaurus

Thesaurus Entry

Copy Paste

Length: [] *

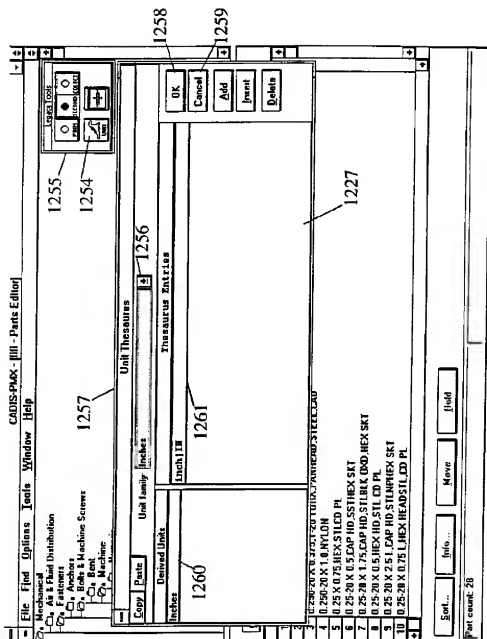
OK Cancel Add Insert Delete

	Description	Feet	Length	Description After Legacy Processing
1	0.25-20 X 0.5625 L.HEX HEAD/STL CD PL			
2	0.25-20 X 2.5 L.HEX HEAD/STL ZINC PLATED			
3	0.25-20 X 0.375 1-20 TORQ PAN/HEAD STEEL CD			
4	0.25-20 X 0.375 1-20 TORQ PAN/HEAD STEEL CD			
5	0.25-20 X 0.375 1-20 TORQ PAN/HEAD STEEL CD			
6	0.25-20 X 0.375 1-20 TORQ PAN/HEAD STEEL CD			
7	0.25-20 X 1.75 CAP 10.5 L.HEX SKT			
8	0.25-20 X 0.5 L.HEX HD STL CD PL			
9	0.25-20 X 2.5 L CAP 10.5 L.HEX SKT			
10	0.25-20 X 0.75 L.HEX HEAD/STL CD PL			

Part count: 26

Get... Info... Move Hold

FIG. 191



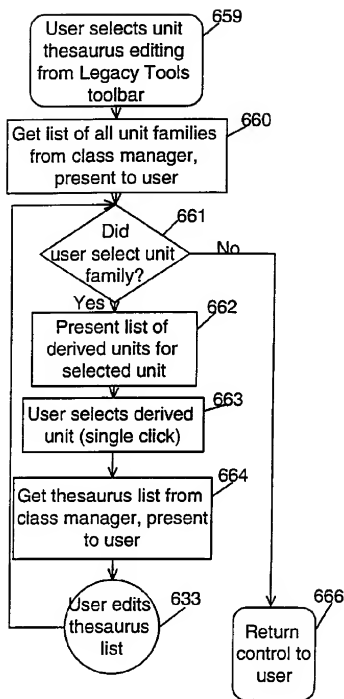


FIG. 193

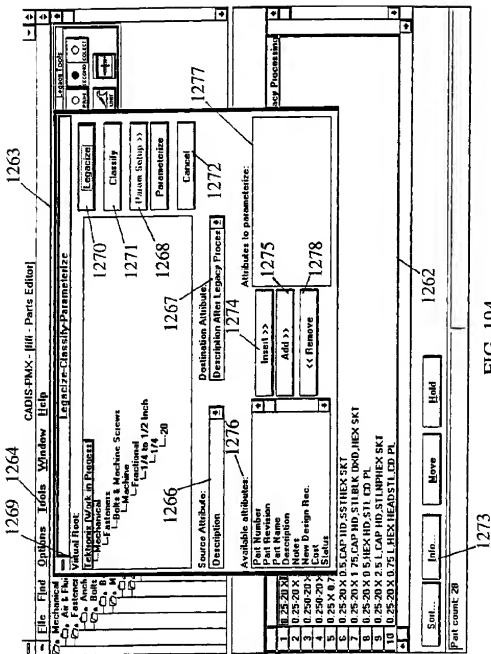


FIG. 194

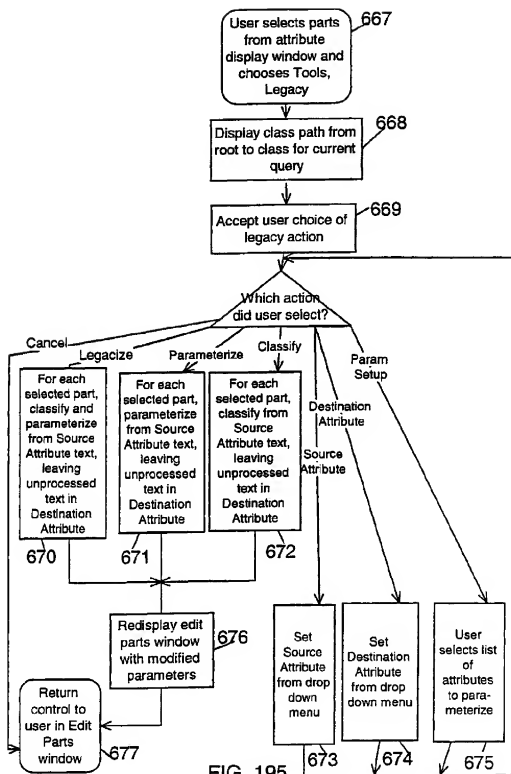
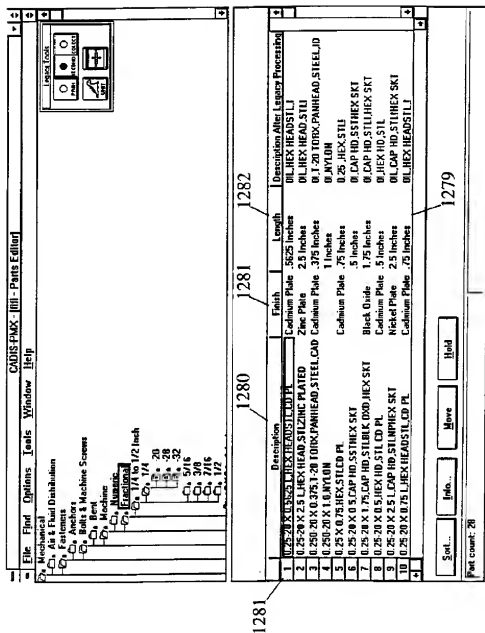


FIG. 195



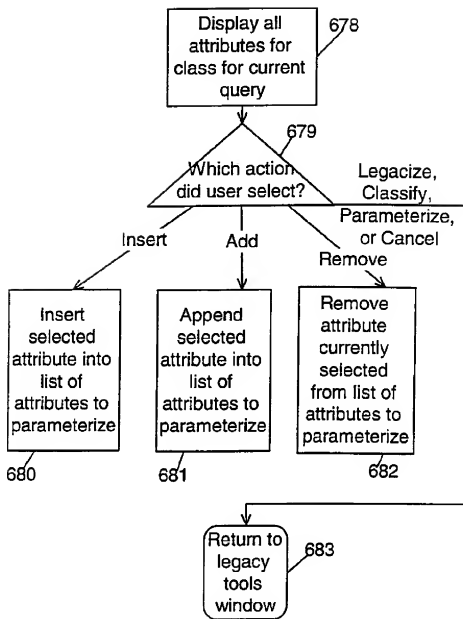


FIG. 197

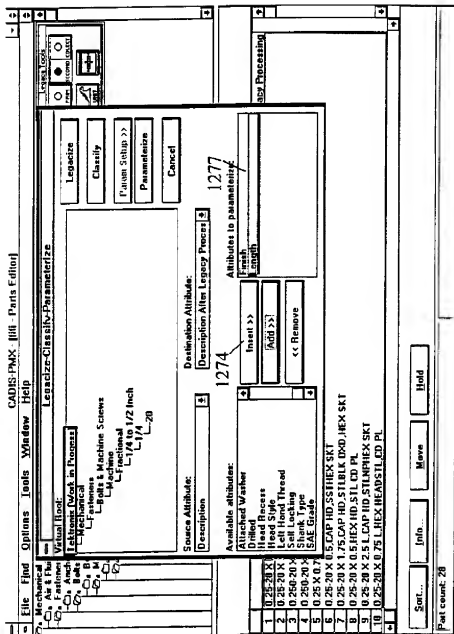


FIG. 198

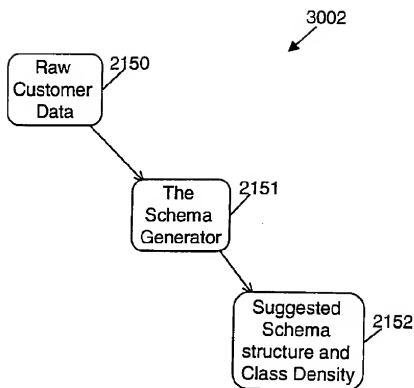


FIG. 199

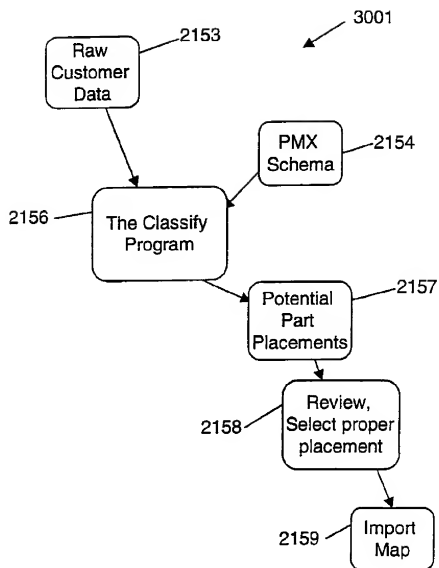


FIG. 200

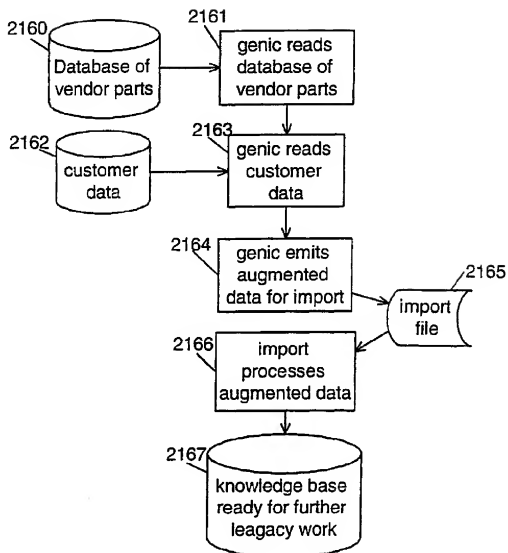


FIG. 201

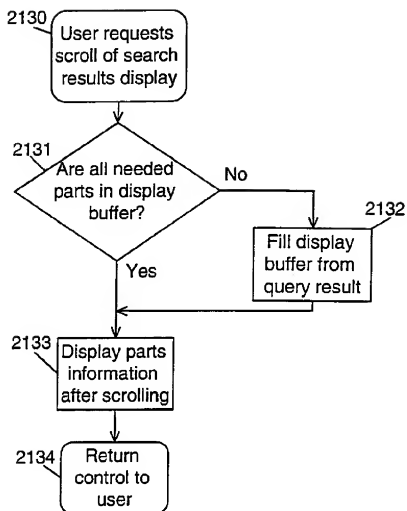


FIG. 202

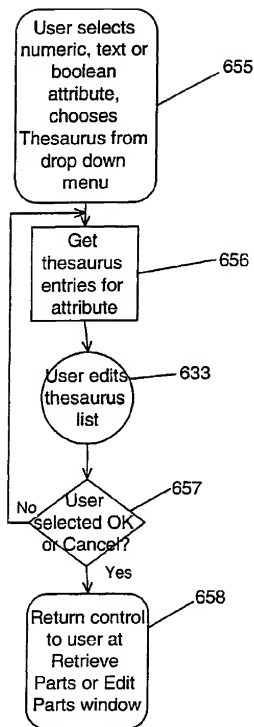


FIG. 203

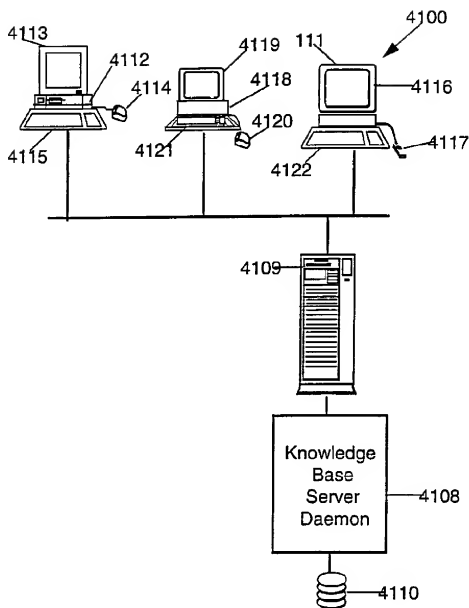


FIG. 204

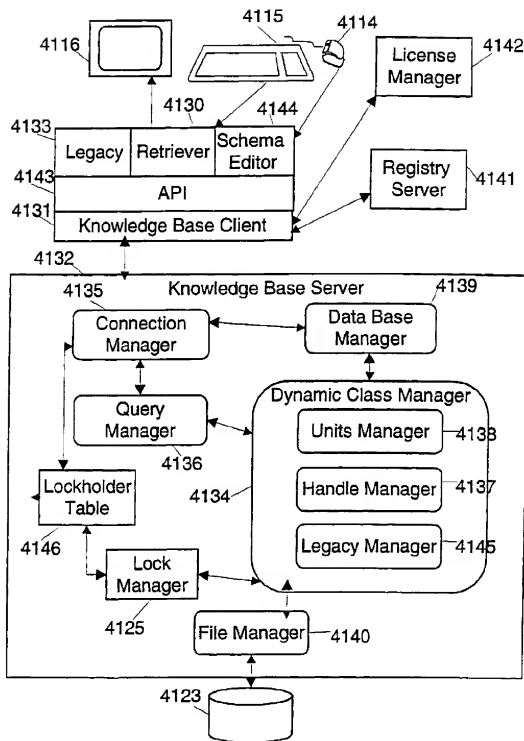


FIG. 205

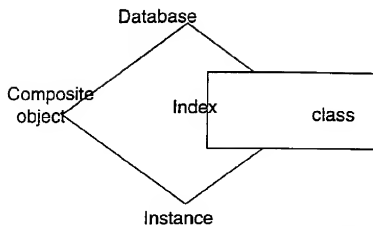


Fig. 206A

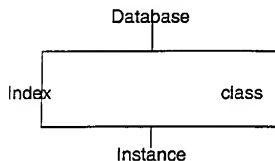


Fig. 206B

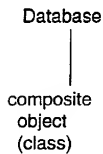


Fig. 206C

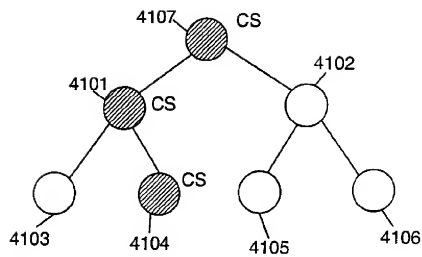


FIG. 207A

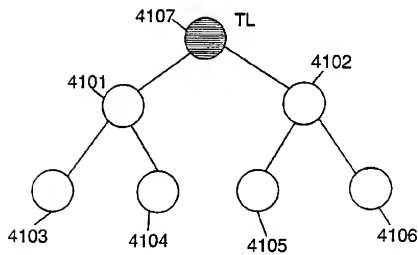


FIG. 207B

【图 208】

4220 4217 Lock Conflict Table 4219

4220	4217	CSL	TSL	TUL	TXL
		No	No	No	Yes
		No	No	No	Yes
4216		No	No	Yes	Yes
		Yes	Yes	Yes	Yes

4221

FIG. 208

【图 209】

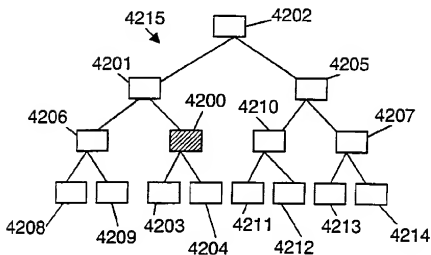


FIG. 209

【图 210】

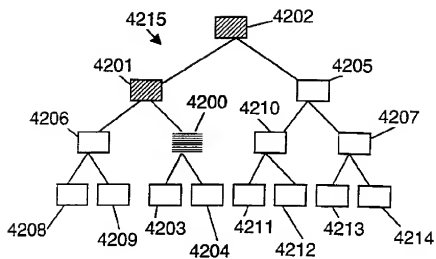


FIG. 210

【图 211】

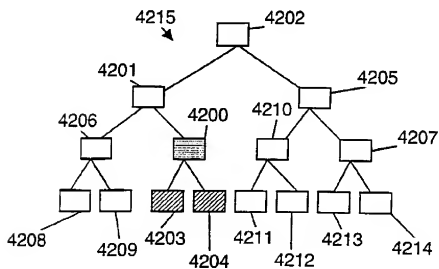


FIG. 211

【图 2 1 2】

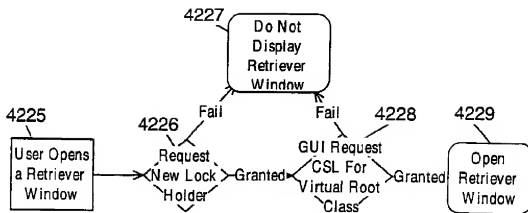


FIG. 212

【图 2 1 3】

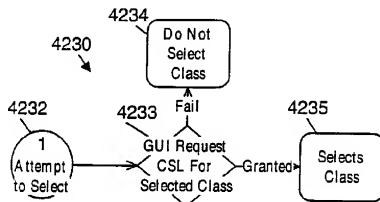


FIG. 213

【图 2 1 4】

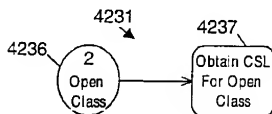


FIG. 214

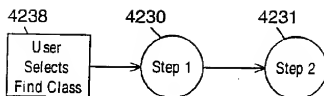


FIG. 215

CAUIS-PMX

File Find Options Tools Window Help

File - Parts Specification

Query Type: Global

Parts Found: 7

Attribute	Search Criteria
1 Vendor Device No.	T
2 Customer Part Number	T
Vendor Name	()
Address	T
Telephone	()
Discount Rate	%

4245 (File icon), 4240 (Find icon), 4241 (Options icon), 4247 (Tools icon), 4248 (Window icon), 4243 (Help icon), 4246 (Display button), 4244 (Edit button), 4242 (Add button), 4290 (Display button), 4290 (Set All button), 4290 (Clear All button), 4290 (Clear Selected button), 4290 (Search Criteria button)

FIG. 216

【图 2 1 7】

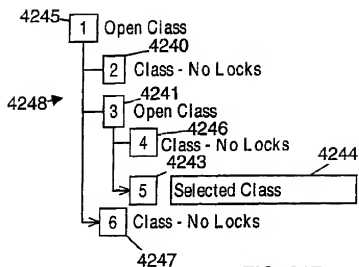


FIG. 217

【图 2 1 8】

	Class	Lock	Lock	Lock	Lock	...
4251	1	4261	CSL			
4252	2	4249				
4253	3	4262	CSL			
4254	4	4269				
4255	5		CSL			
	...					

4250 points to the header row.

4256 points to the 'Class' column.

4257 points to the 'Lock' column.

4258 points to the 'Lock' column.

4259 points to the 'Lock' column.

4260 points to the 'Lock' column.

FIG. 218

【图 2 1 9】

Lock Object		
Lock Type	Count	
TXL - Tree Exclusive Lock	0	4264
TUL - Tree Update Lock	0	4265
TSL - Tree Share Lock	0	4266
CSL - Class Share Lock	1	4263
User ID	100	4268
Lock Holder Handle	1	4267

FIG. 219

【图 2 2 0】

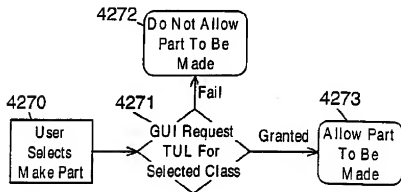


FIG. 220

【图 2 2 5】

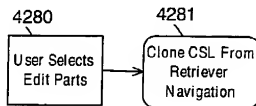


FIG. 225

【图 2 2 6】

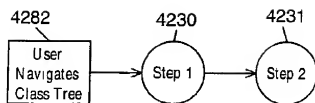


FIG. 226

【图 2 2 1】

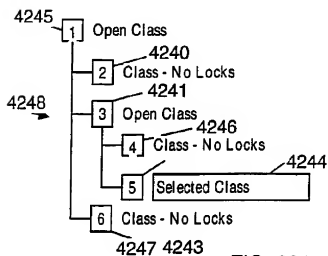


FIG. 221

【图 2 2 2】

4250 ↓

	4256	4257	Lock Table 4258	4259	
4251	Class	Lock	Lock	Lock	...
4252	1	4261	CSL		
4253	2	4249			
4254	3	4262	CSL		
4255	4	4269			
	5	4260	CSL,TUL		
	...				

FIG. 222

4260

Lock Type	Count
TXL - Tree Exclusive Lock	0
TUL - Tree Update Lock	1
TSL - Tree Share Lock	0
CSL - Class Share Lock	1
User ID	100
Lock Holder Handle	1

4264
4265
4266
4263
4268
4267

FIG. 223

4290

CADIS PMX

File Find Options Tools Window Help

fill - Parts Specification

Parts Found 0

1 2 3 4 5 6

1 Vendor Device No. T

2 Customer Part Number T

Vendor Name (S)

Search Criteria

4245 4241 4276 4243

Add Part

4248

4241

4243

4275

Display Edit

Attributes 3/g Values

Discard Rate

Add Cancel

FIG. 224

CADIS-PMX

File Find Options Window Help

Parts Found: 7

Query type: Global

Attribute

Vendor Device No.

Search Criteria

III - Parts Editor

4290

4248

4285

Display

Defining Class is Class 5

Vendor Device No.	Customer Part Number
1/Acme	Defining Class is Class 5
2/Acme	Defining Class is Class 5
3/Acme	Defining Class is Class 5
4/Acme	Defining Class is Class 5
5/Acme	Defining Class is Class 5
6/Acme	Defining Class is Class 5
7/Acme	Defining Class is Class 5

Sort... Info... Move Delete

Part count: 7

4283

FIG. 227

【图 2 2 8】

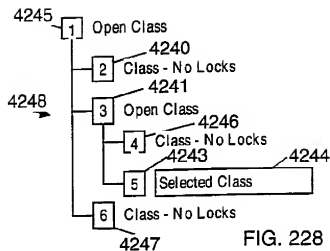


FIG. 228

【图 2 2 9】

	4256	4257	Lock Table	4258	4259
4251	Class	Lock	Lock	Lock	Lock
4252	1	4261	CSL, CSL		
4253	2	4249			
4254	3	4262	CSL, CSL		
4255	4	4269			
	5	4260	CSL, CSL		
	...				

FIG. 229

【图 2 3 0】

4260	Lock Type	Count
	TXL - Tree Exclusive Lock	0
	TUL - Tree Update Lock	0
	TSL - Tree Share Lock	0
	CSL - Class Share Lock	2
	User ID	100
	Lock Holder Handle	1

FIG. 230

【图 2 3 1】

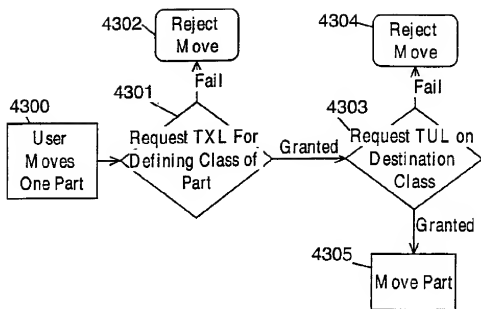


FIG. 231

【图 2 3 2】

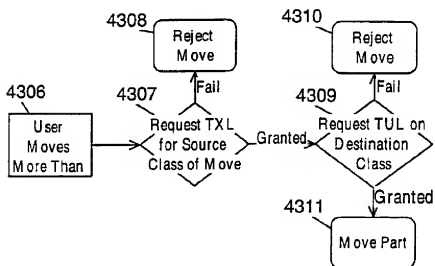


FIG. 232

【図 2 3 3】

	Class	Lock Object	Lock Object	Lock Object	Lock Object	...
4251	1		CSL, CSL	4261		
4252	2			4249		
4253	3		CSL, CSL, TUL	4262		
4254	4			4269		
4255	5		CSL, CSL, TXL			
	...					

4256 4257 4258 4259

4260

4250

FIG. 233

【図 2 3 4】

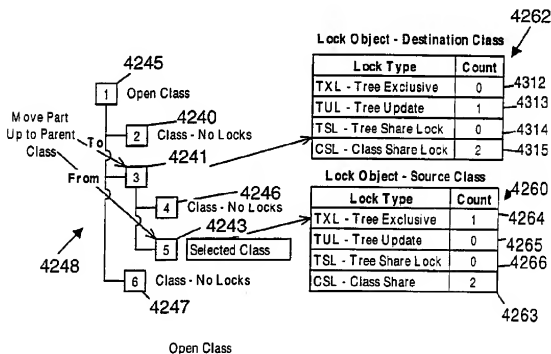


FIG. 234

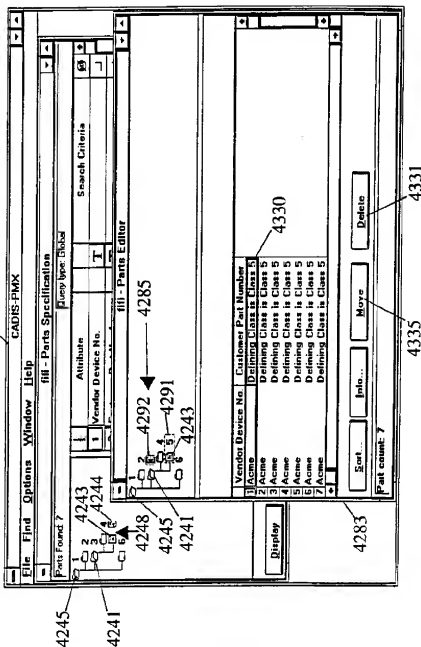


FIG. 235

【图 2 3 6】

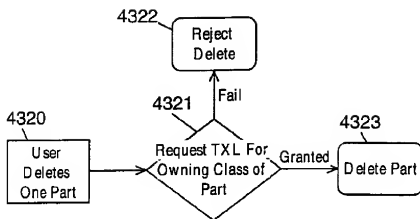


FIG. 236

【图 2 3 7】

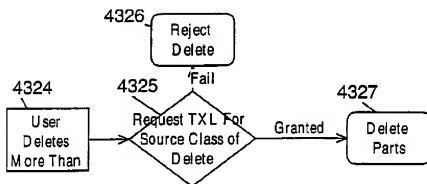


FIG. 237

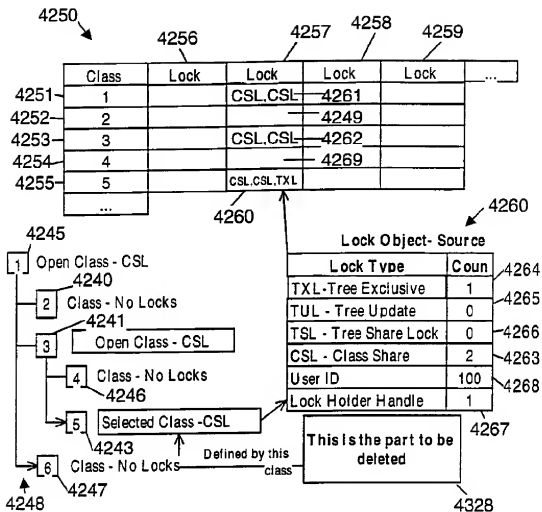


FIG. 238

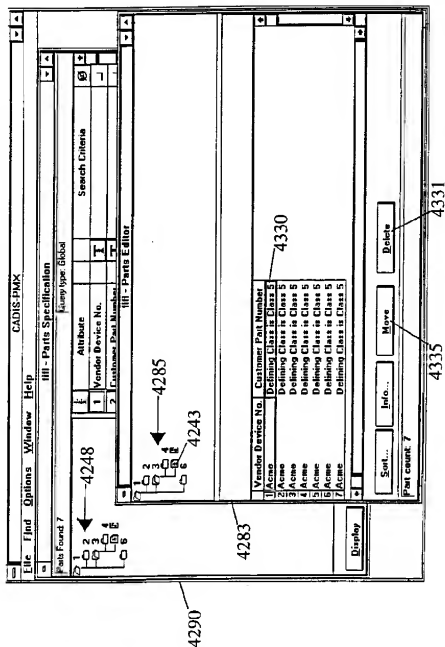


FIG. 239

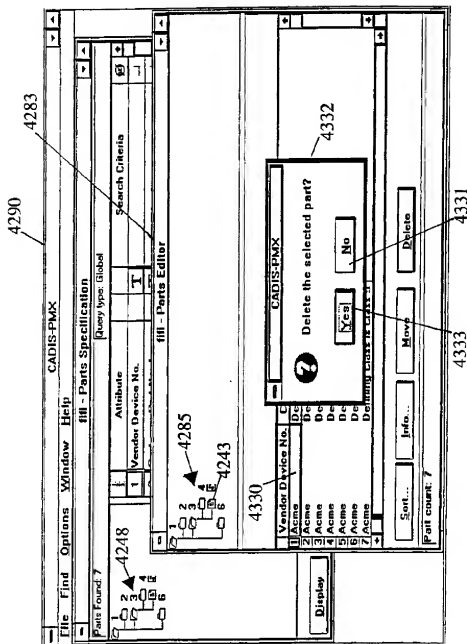


FIG. 240

【图 2 4 1】

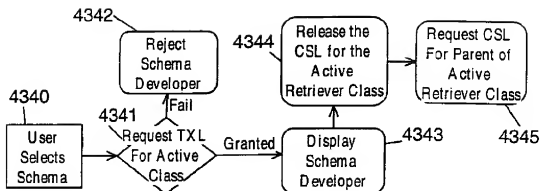


FIG. 241

【图 2 4 2】

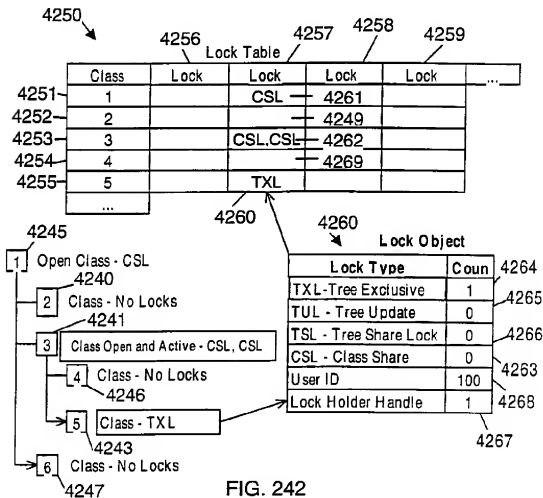


FIG. 242

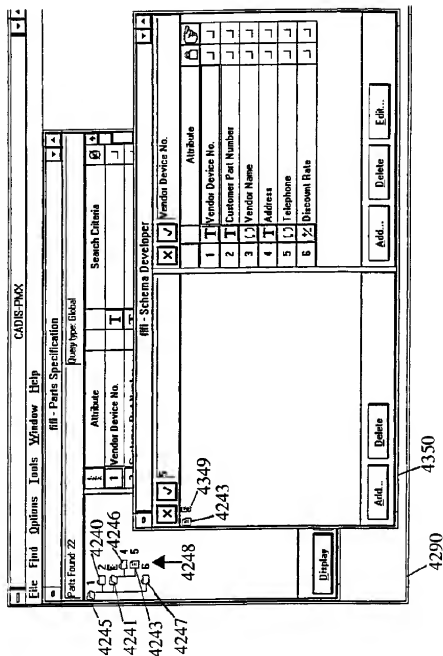


FIG. 243

【图 2 4 4】

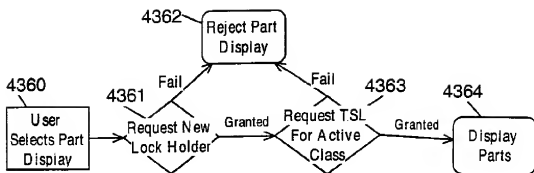


FIG. 244

【图 2 4 5】

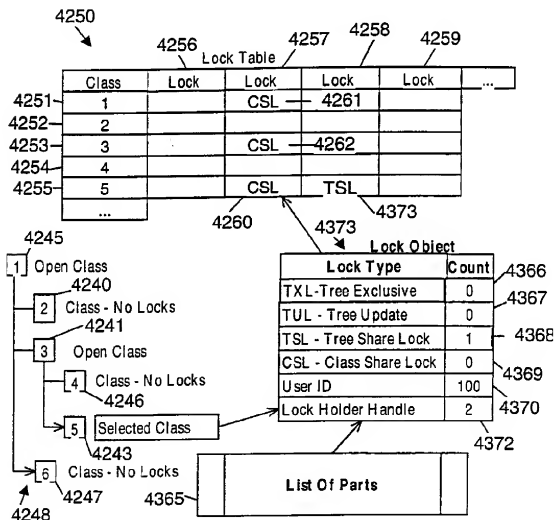


FIG. 245

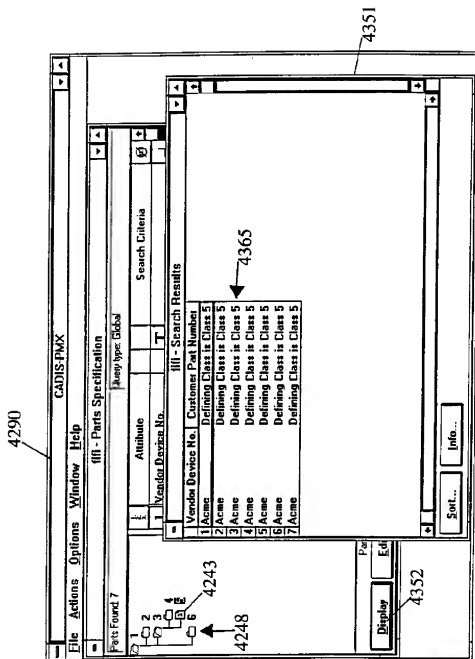


FIG. 246

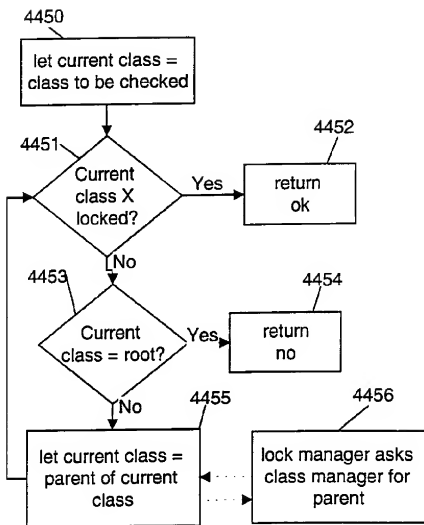


FIG. 247

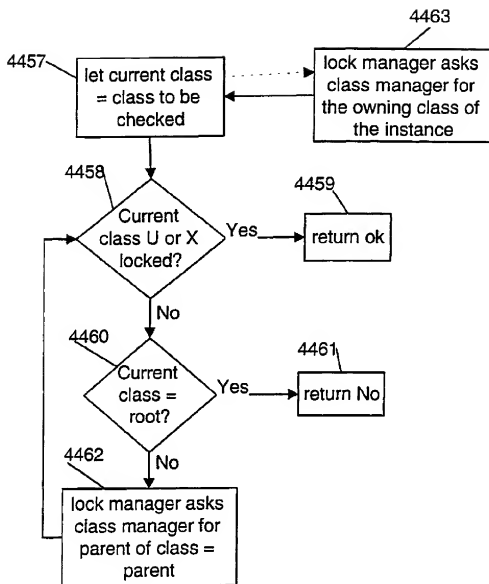


FIG. 248

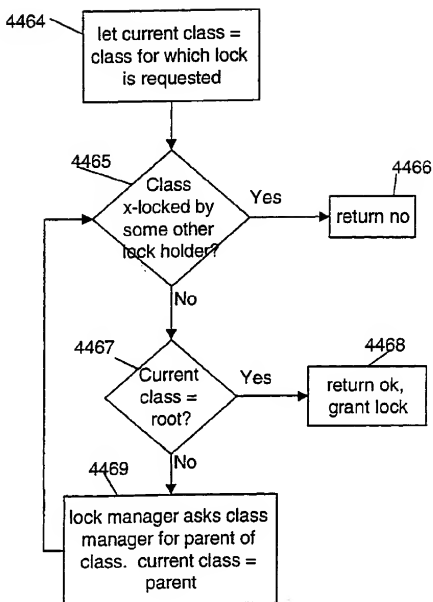


FIG. 249

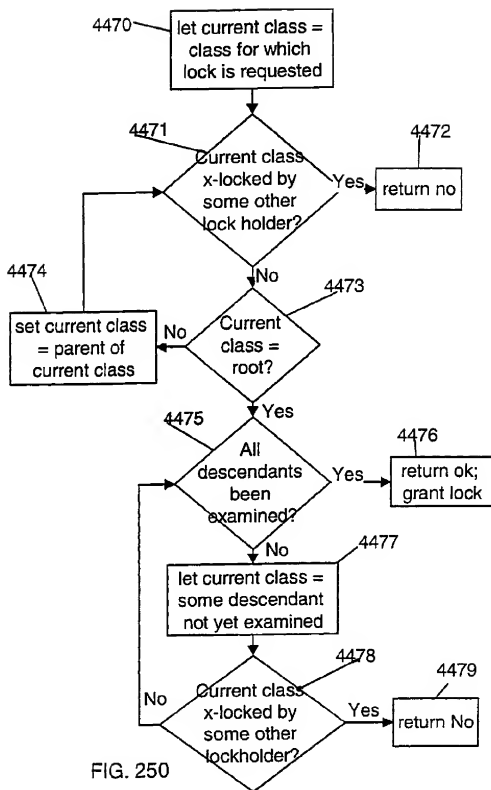
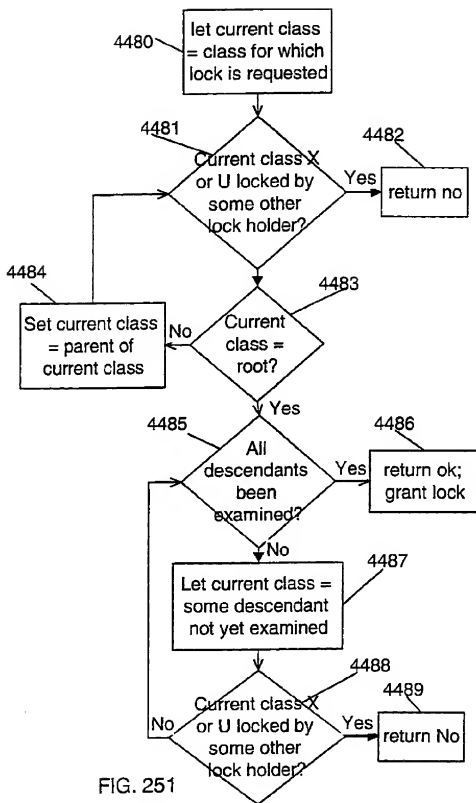


FIG. 250



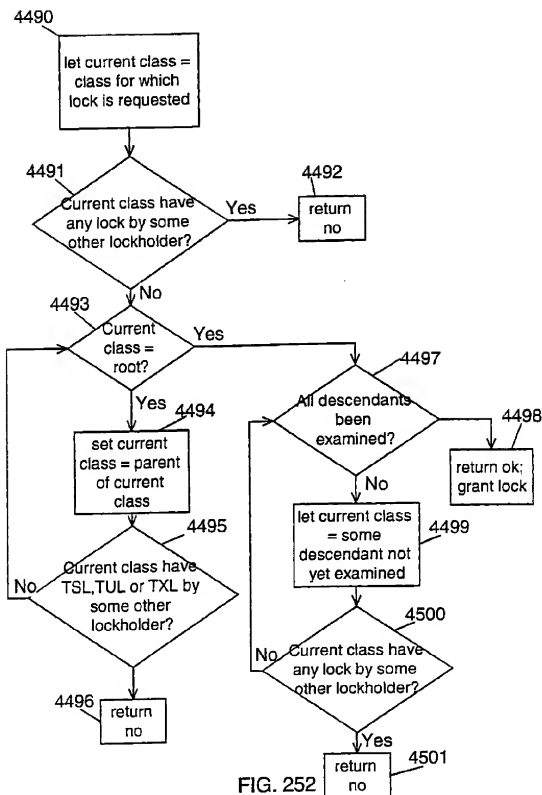


FIG. 252

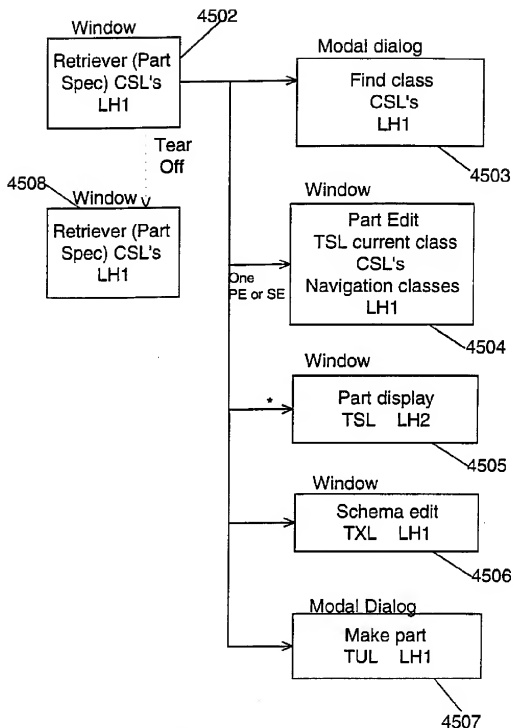


FIG. 253

【图 2 5 4】

	LOCK HOLDER 1	LOCK HOLDER 2	LOCK HOLDER 3	LOCK HOLDER 4	LOCK HOLDER 5	LOCK HOLDER 6	LOCK HOLDER 7
CLASS HANDLE 1	<u>4401</u>	<u>4408</u>	<u>4415</u>	<u>4422</u>	<u>4429</u>	<u>4436</u>	<u>4443</u>
CLASS HANDLE 2	<u>4402</u>	<u>4409</u>	<u>4416</u>	<u>4423</u>	<u>4430</u>	<u>4437</u>	<u>4444</u>
CLASS HANDLE 3	<u>4403</u>	<u>4410</u>	<u>4417</u>	<u>4424</u>	<u>4431</u>	<u>4438</u>	<u>4445</u>
CLASS HANDLE 4	<u>4404</u>	<u>4411</u>	<u>4418</u>	<u>4425</u>	<u>4432</u>	<u>4439</u>	<u>4446</u>
CLASS HANDLE 5	<u>4405</u>	<u>4412</u>	<u>4419</u>	<u>4426</u>	<u>4433</u>	<u>4440</u>	<u>4447</u>
CLASS HANDLE 6	<u>4406</u>	<u>4413</u>	<u>4420</u>	<u>4427</u>	<u>4434</u>	<u>4441</u>	<u>4448</u>
CLASS HANDLE 7	<u>4407</u>	<u>4414</u>	<u>4421</u>	<u>4428</u>	<u>4435</u>	<u>4442</u>	<u>4449</u>

4400
↗

FIG. 254

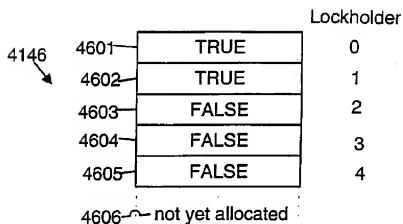


FIG. 255

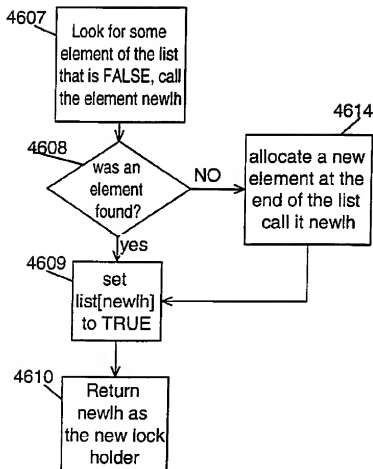


FIG. 256

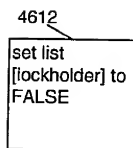


FIG. 257

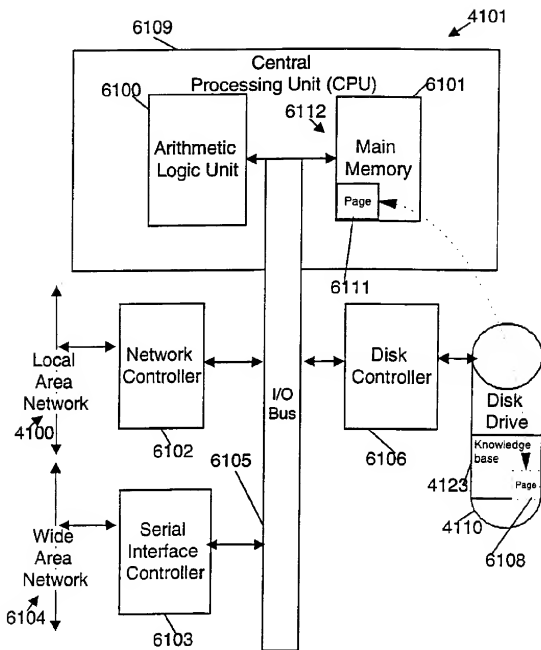


FIG. 258

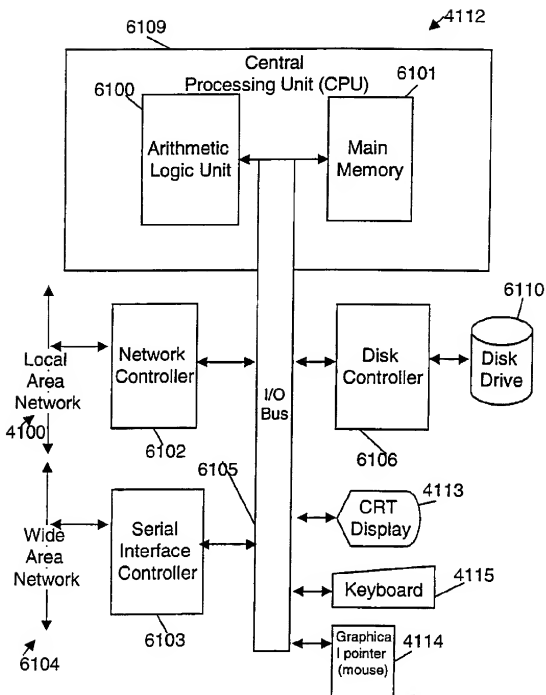


FIG. 259

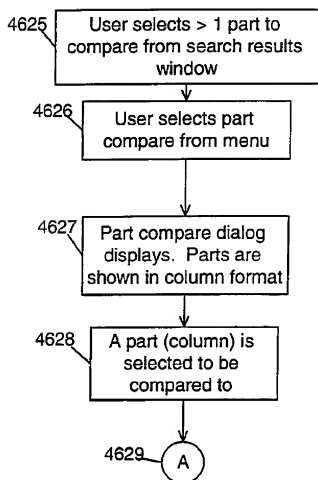


FIG. 260

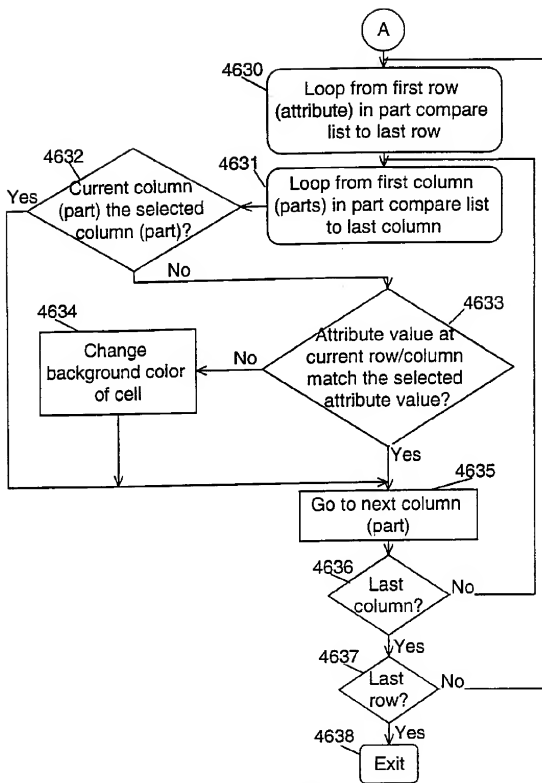


FIG. 261



File	Actions	Options	Window	Help	LAURIS-PMX			
					Item test - Search Results	Standard Name	Data Classification System Code	
Item Part	Compare Parts...	Item Description	Part	Item Description	Part	Standard Name	Data Classification System Code	
1	15825840	MIDDLE	EPDM 32K 00D	VF-MOSHOD	23306	VF-MOSHOD	23306	4653
2	5582942	MIDDLE	EPDM 32K EVEN	VF-MOSHOD	23306	VF-MOSHOD	23306	4654
3	34587358	MODULE	EPDM 32K 00D	VF-MOSHOD	23306	VF-MOSHOD	23306	4655
4	15582940	MODULE	EPDM 32K EVEN	VF-MOSHOD	23306	VF-MOSHOD	23306	4656
5	5568746	EPROM	80C8 EPROM 120K5 501C-28	VF-MOSHOD	23306	VF-MOSHOD	23306	4657
6	6212424	EPROM	DO NOT USE PART. NEW APPLICATIONS NOT SUPPORTED.	VF-MOSHOD	23306	VF-MOSHOD	23306	4658
7	6231572	EPROM	DO NOT USE PART. IS NOT AVAILABLE	VF-MOSHOD	23306	VF-MOSHOD	23306	4659
8	6231572	EPROM	DO NOT USE PART. NEW APPLICATIONS NOT SUPPORTED.	VF-MOSHOD	23306	VF-MOSHOD	23306	4660
9	6231574	EPROM	DO NOT USE PART. NEW APPLICATIONS NOT SUPPORTED.	VF-MOSHOD	23306	VF-MOSHOD	23306	4661
10	6266040	EPROM	1K SERIAL EPROM (DIP10), 1000NS, 501C-8	VF-MOSHOD	23306	VF-MOSHOD	23306	4662
11	116413113	NPVRAM	256 X 4 30NS NPVRAM	VF-MOSHOD	23306	VF-MOSHOD	23306	4663
12	16456363	OIP	DO NOT USE PART. NEW APPLICATIONS NOT SUPPORTED.	VF-MOSHOD	23306	VF-MOSHOD	23306	4664
13	58255512	OIP	DO NOT USE PART. PART IS NOT AVAILABLE.	VF-MOSHOD	23306	VF-MOSHOD	23306	
14	58255513	OIP	DO NOT USE PART. PART IS NOT AVAILABLE.	VF-MOSHOD	23306	VF-MOSHOD	23306	
15	58255514	OIP	DO NOT USE PART. PART IS NOT AVAILABLE.	VF-MOSHOD	23306	VF-MOSHOD	23306	
16	58255515	OIP	12K DIP/PM (CSINS, 100)	VF-MOSHOD	23306	VF-MOSHOD	23306	
17	58255516	OIP	DO NOT USE PART. NEW APPLICATIONS NOT SUPPORTED.	VF-MOSHOD	23306	VF-MOSHOD	23306	
18	58255517	OIP	DO NOT USE PART. IS NOT AVAILABLE	VF-MOSHOD	23306	VF-MOSHOD	23306	
19	58255589	EPROM	DO NOT USE PART. IS NOT AVAILABLE.	VF-MOSHOD	23306	VF-MOSHOD	23306	
20	58255752	EPROM	DO NOT USE PART. NEW APPLICATIONS NOT SUPPORTED.	VF-MOSHOD	23306	VF-MOSHOD	23306	

FIG. 262A

4630

4638

4633

4634

4645

4635

4636

4637

4631

4632

4648

4644

4647

4643

4642

4641

4640

4639

4646

4647

4641

4646

4641

4646

4641

4646

4641

4646

4641

4646

4641

4646

4641

4646

4641

4646

4641

4646

4641

4646

4641

4646

4641

4646

4641

4646

4641

4646

4641

4646

4641

4646

4641

4646

4641

4646

4641

4646

4641

4646

4641

4646

4641

4646

4641

4646

4641

4646

4641

4646

4641

4646

4641

4646

4641

4646

4641

4646

4641

4646

4641

4646

4641

4646

4641

4646

4641

4646

4641

4646

4641

4646

4641

4646

4641

4646

4641

4646

4641

4646

4641

4646

4641

4646

4641

4646

4641

4646

4641

4646

4641

4646

4641

4646

4641

4646

4641

4646

4641

4646

4641

4646

4641

4646

4641

4646

4641

4646

4641

4646

4641

4646

4641

4646

4641

4646

4641

4646

4641

4646

4641

4646

4641

4646

4641

4646

4641

4646

4641

4646

4641

4646

4641

4646

4641

4646

4641

4646

4641

4646

4641

4646

4641

4646

4641

4646

4641

4646

4641

4646

4641

4646

4641

4646

4641

4646

4641

4646

4641

4646

4641

4646

4641

4646

4641

4646

4641

4646

4641

4646

4641

4646

4641

4646

4641

4646

4641

4646

4641

FIG. 262B

Part Attribute Comparison				
Attribute Title	Part 1	Part 2	Part 3	Part 4
Part Number	213104370	21310535A	213104370	21437435A
Basic Part Name	SCR ASS'Y WSHR	SCREW MACHINE	SCREW MACHINE	SCREW
Associated File Name	c:/pwr/mr/200732.dwg	c:/pwr/mr/200731.dwg	c:/pwr/mr/200732.dwg	c:/pwr/mr/200733.dwg
= Cost				
Finish	Cadmium Plate	Zinc Plate	Cadmium Plate	Nickel
Major Material	Steel	Steel	Steel	
Attached Washer				
= Drilled				
Head Recess			Text	
Head Style	Hex	Hex	Pan	
= Left Hand Thread				
= Self Locking				
= Shank Type				
Length	5.625 inches	2.5 inches	.375 inches	1 inches
= Self Grade				

Compare To Selected Part

Clear Comparisons

Legend: Match No Match

Done

FIG. 263

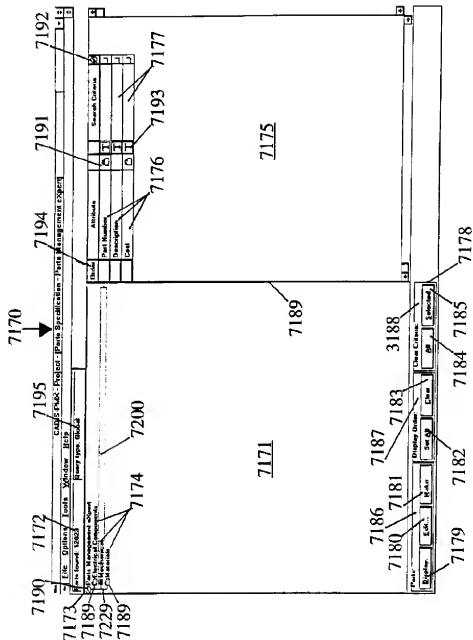


FIG. 264

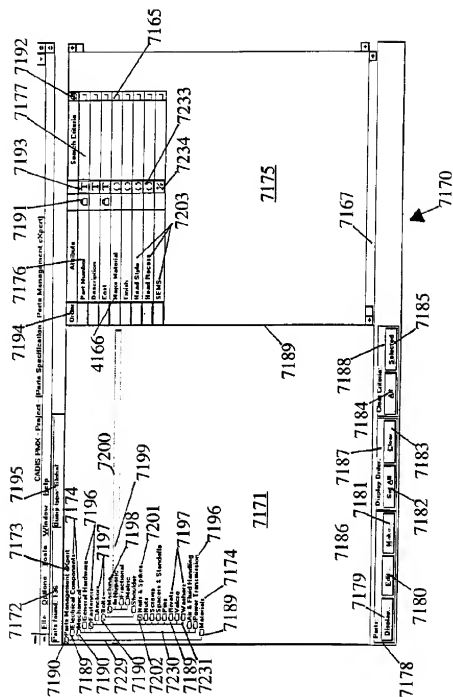


FIG. 265

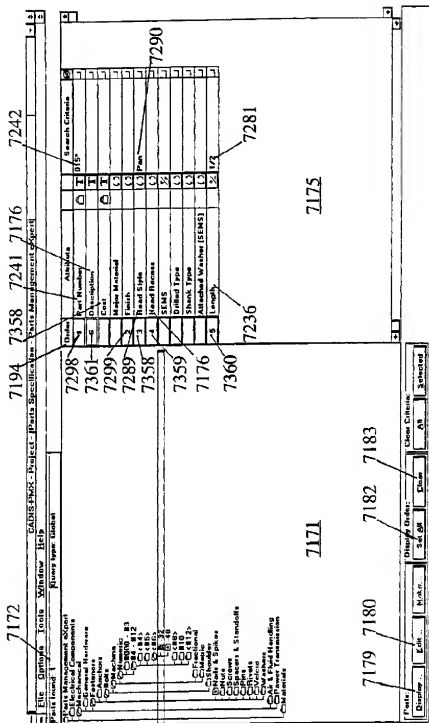


FIG. 266

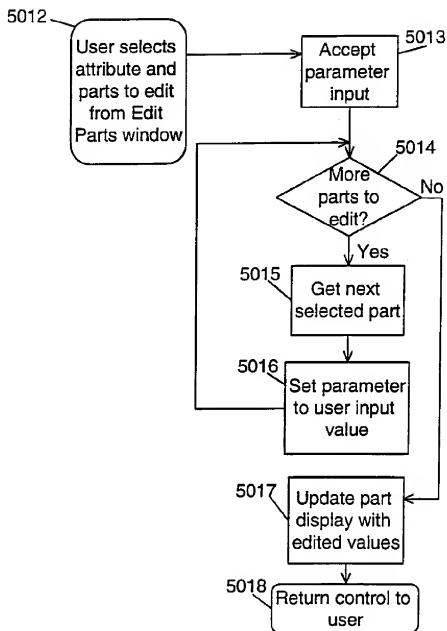


FIG. 267

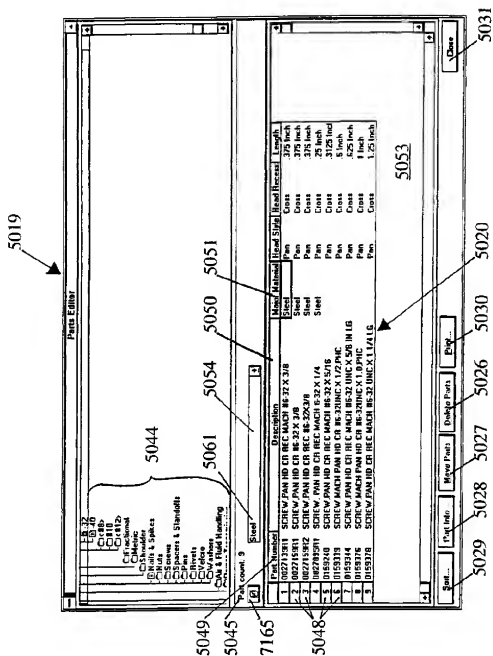


FIG. 268

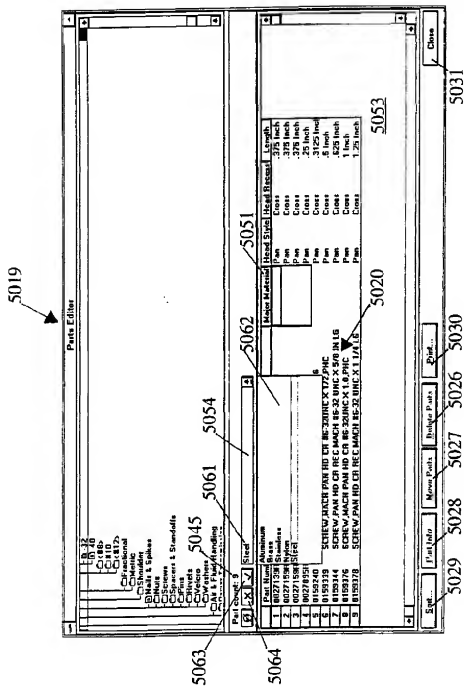


FIG. 269

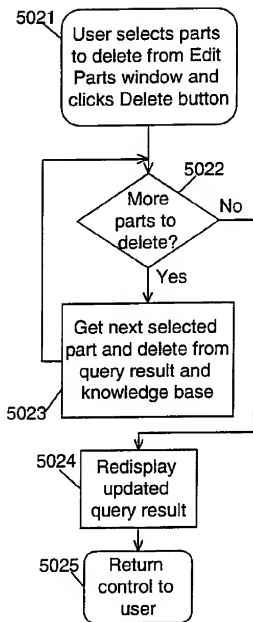


FIG. 270

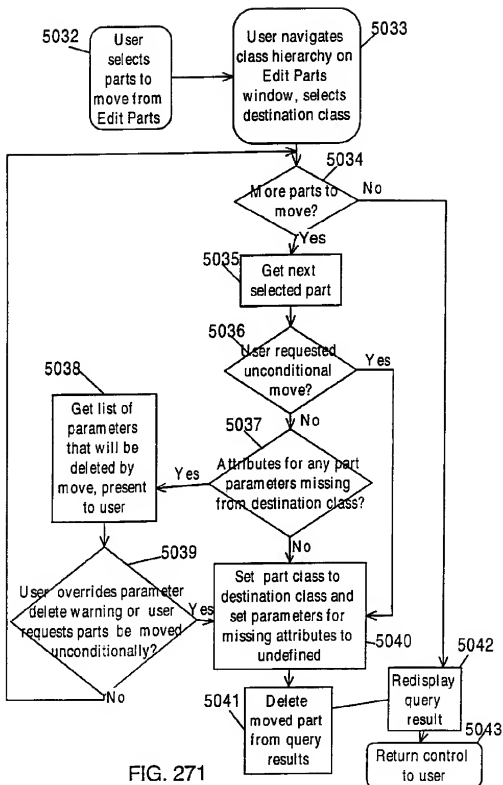
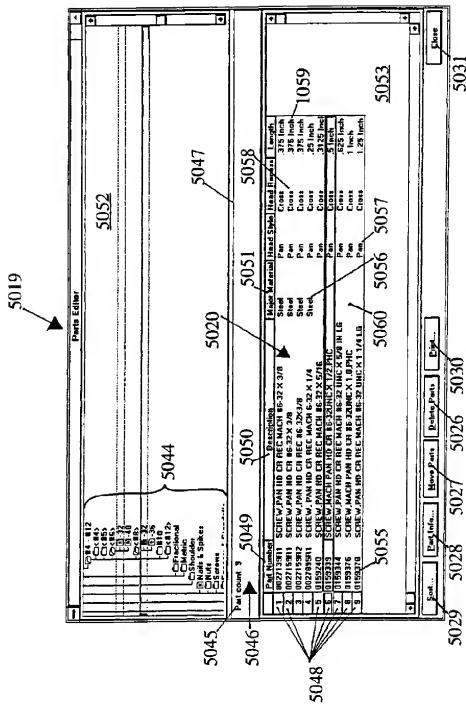


FIG. 271



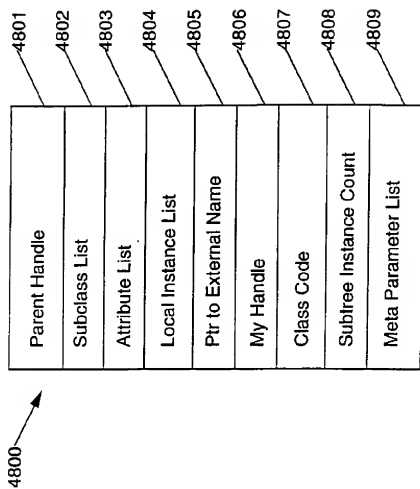


FIG. 273

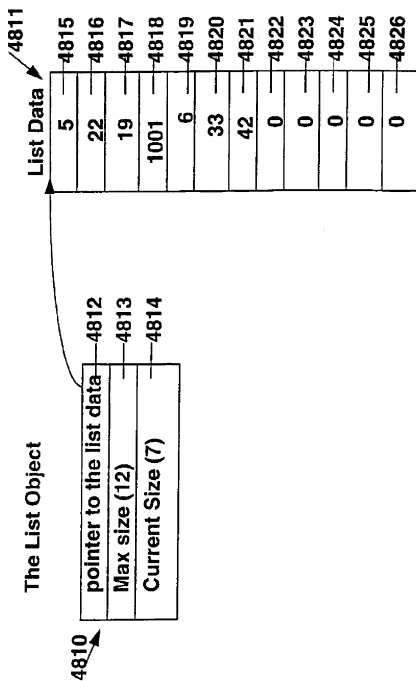


FIG. 274

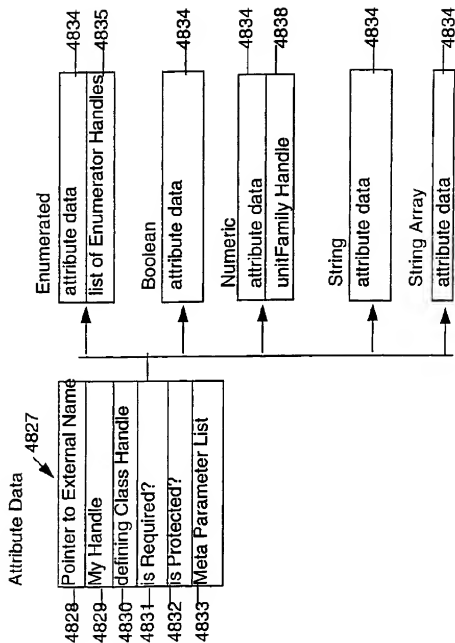


FIG. 275

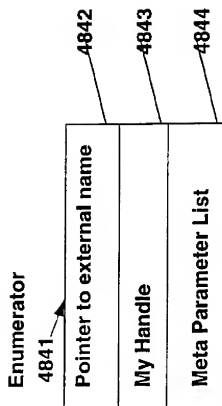


FIG. 276

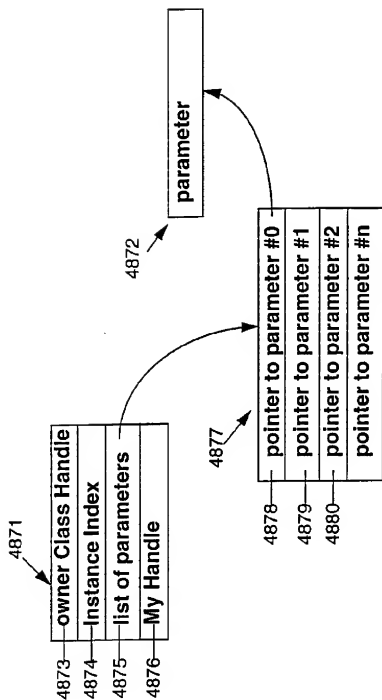


FIG. 277

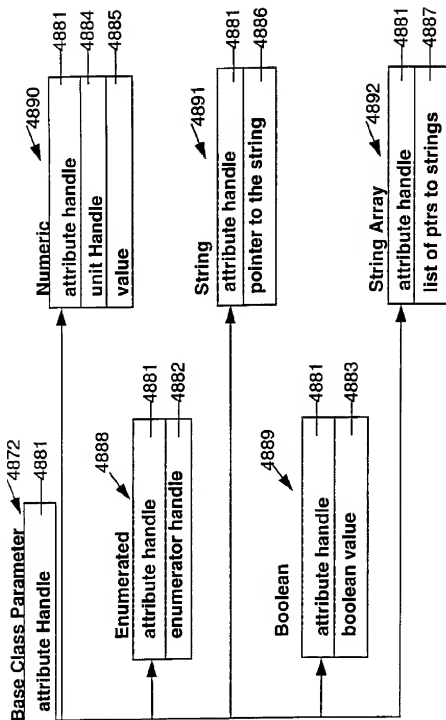


FIG. 278

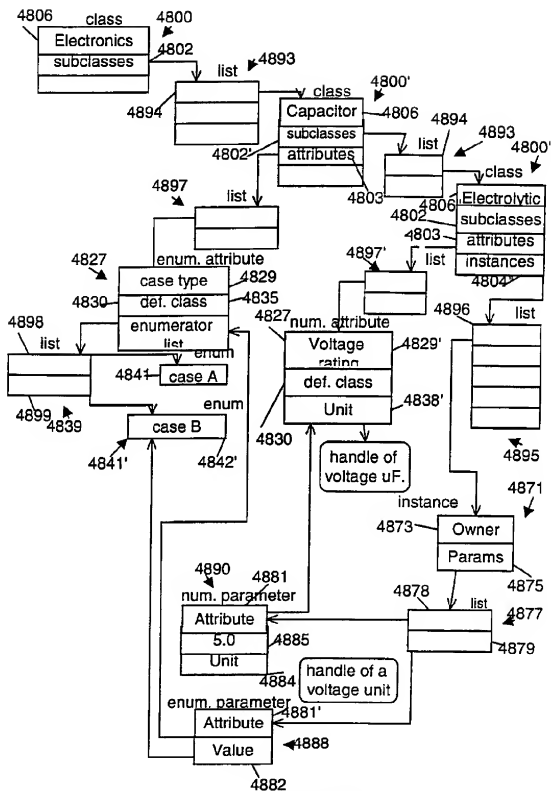


FIG. 279

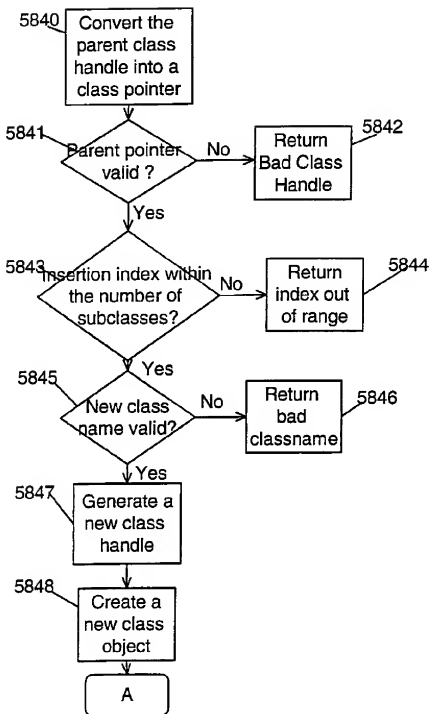


FIG. 280

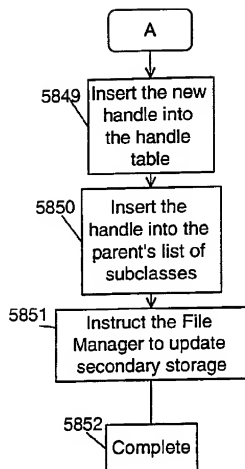


FIG. 281

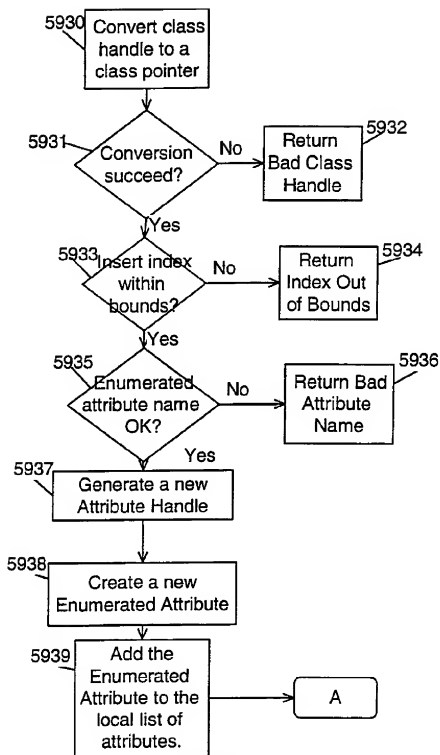


FIG. 282

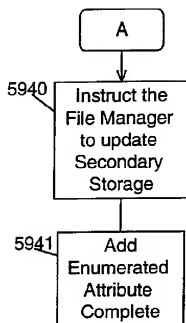


FIG. 283

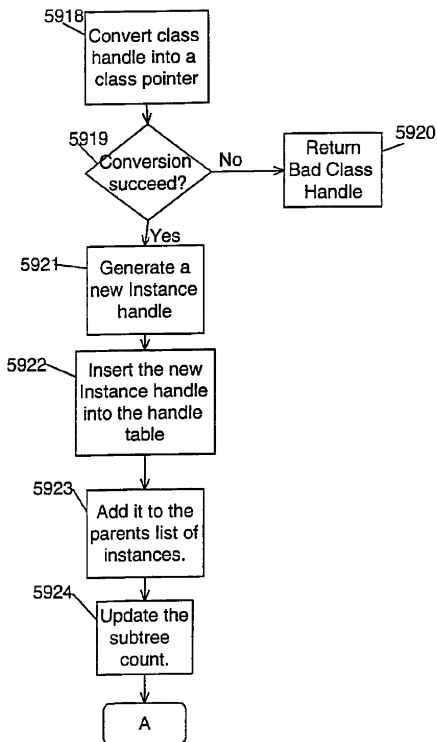


FIG. 284

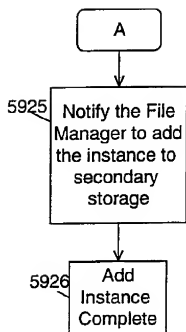


FIG. 285

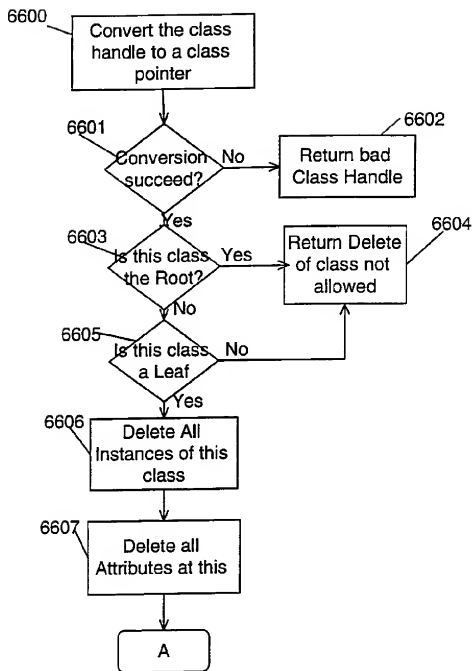


FIG. 286

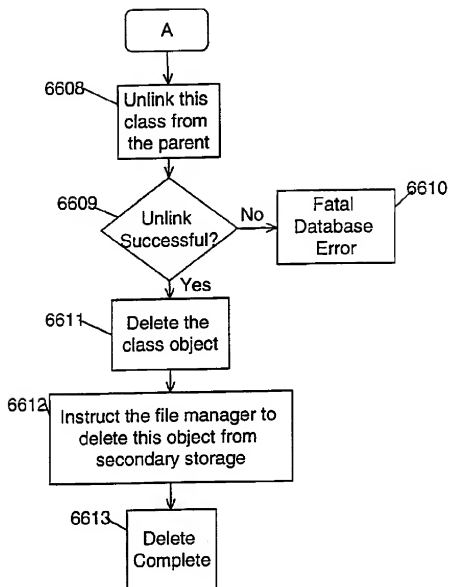


FIG. 287

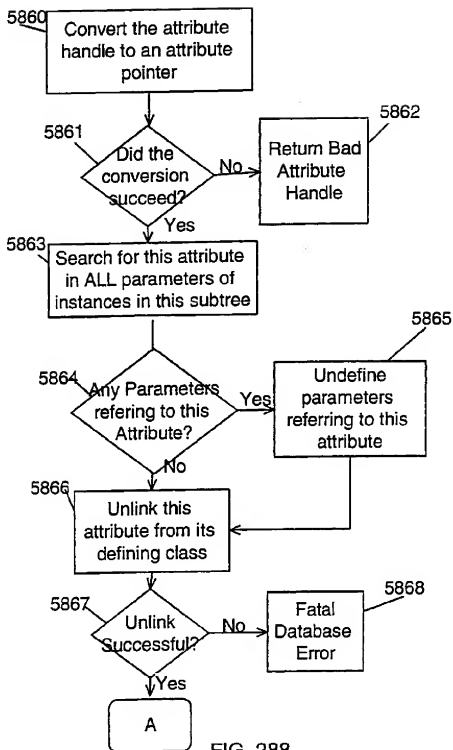


FIG. 288

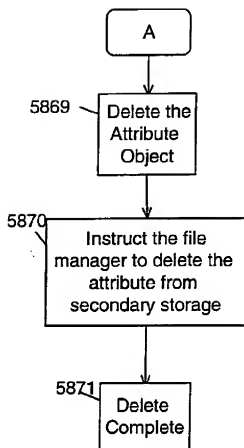


FIG. 289

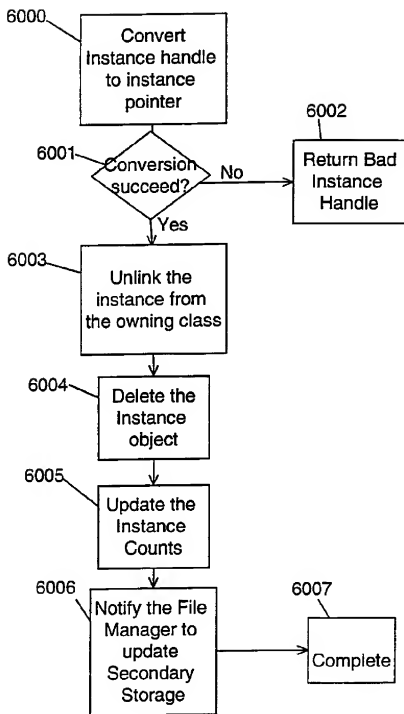


FIG. 290

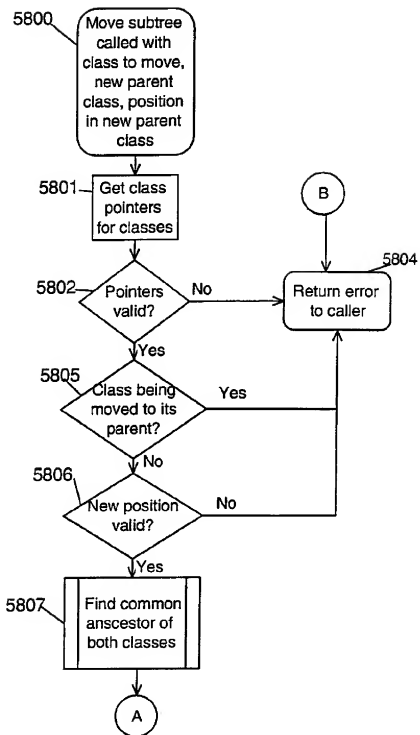


Fig. 291

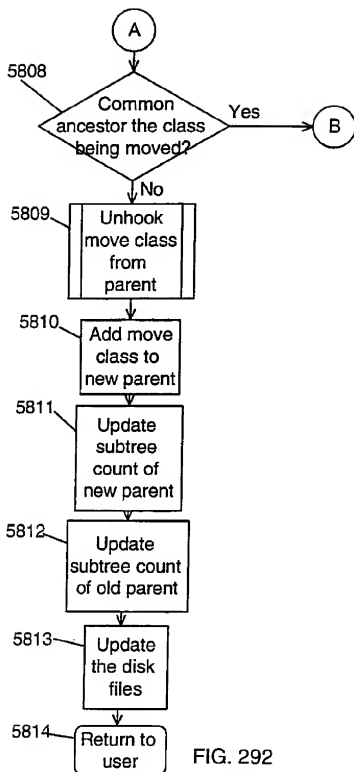


FIG. 292

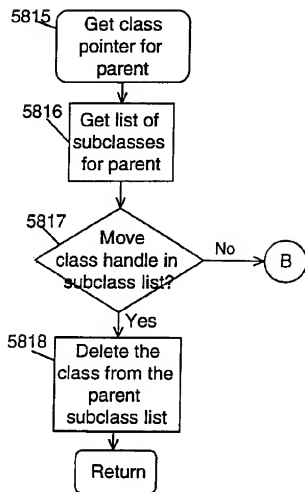


FIG. 293

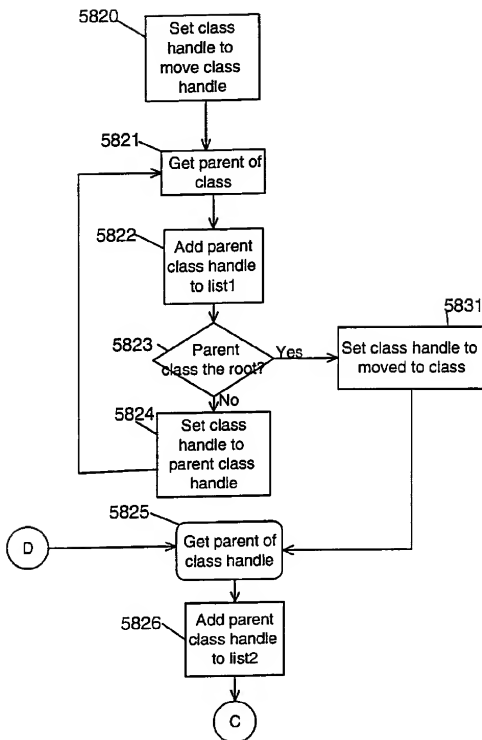


FIG. 294

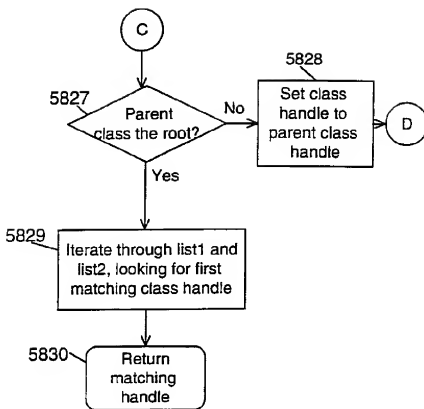


FIG. 295

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US95/15028

A. CLASSIFICATION OF SUBJECT MATTER IPC(6) : G06F 17/30 US CL : 395/600 According to International Patent Classification (IPC) or to both national classification and IPC		
B. FIELDS SEARCHED Minimum documentation searched (classification system followed by classification symbols) U.S. : 395/600, 700 Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched Electronic data base consulted during the international search (name of data base and, where practicable, search terms used) APS search terms object oriented, database, knowledge base, handle, instance, class, attributes, legacy		
C. DOCUMENTS CONSIDERED TO BE RELEVANT		
Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US, A, 4,930,071 (TOU ET AL) 29 MAY 1990	
A	US, A, 5,133,075 (RISCH) 21 JULY 1992	
A	US, A, 5,021,992 (KONDO) 04 JUNE 1991	
<input type="checkbox"/> Further documents are listed in the continuation of Box C. <input type="checkbox"/> See parent family annex.		
* Special categories of cited documents: "A" document defining the general state of the art which is not considered to be part of particular relevance "E" earlier document published on or after the international filing date "L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified) "O" document referring to an oral disclosure, use, exhibition or other means "P" document published prior to the international filing date but later than the priority date claimed	"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention "X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step where the document is taken alone "Y" (combination of particular relevance); the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, each combination being obvious to a person skilled in the art "Z" document member of the same patent family	
Date of the actual completion of the international search 17 APRIL 1996		Date of mailing of the international search report 22 APR 1996
Name and mailing address of the ISA/US Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231 Facsimile No.: (703) 305-2236		Authorized officer JOHN C. LOOMIS Telephone No.: (703) 305-3333

Form PCT/ISA/210 (second sheet)(July 1992)*

(81)指定国 EP(AT, BE, CH, DE, DK, ES, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OA(BF, BJ, CF, CG, CI, CM, GA, GN, ML, MR, NE, SN, TD, TG), AT, AU, BB, BG, BR, BY, CA, CH, CN, CZ, DE, DK, ES, FI, GB, HU, JP, KP, KR, KZ, LK, LU, LV, MG, MN, MW, NO, NZ, PL, PT, RO, RU, SD, SE, SK, UA, UZ, VN

(72)発明者 ハイニ, ウィリアム・シー
アメリカ合衆国カラドウ州80005、アー
ヴェイダ、ジャンズン・コート 8256番

(72)発明者 マティカ, ジェン・ディー
アメリカ合衆国カラドウ州80439、エヴ
ァグリーン、チェスナット・ドライヴ
30130番

(72)発明者 ベンドルタン, サミュエル・エス
アメリカ合衆国カラドウ州80027、ルイ
スヴィル、ウエスト・ダリア・ストリート
976番

(72)発明者 スモールウッド, タマス・ディー
アメリカ合衆国カラドウ州80026、ラー
フィエト、モーニング・スター・レイ
ン 308番

(72)発明者 ターベニング, ブルック・イー
アメリカ合衆国カラドウ州80403、ゴー
ルドン、ウエストリッジ・ロウド 25221
番

(72)発明者 トラウト, ケニス・エイ
アメリカ合衆国カラドウ州80303、ボウ
ルダ、クーパー・コート 4151番